Pavel Brazdil
Alípio Jorge (Eds.)

LNAI 2258

Progress in Artificial Intelligence

Knowledge Extraction, Multi-agent Systems, Logic Programming and Constraint Solving

10th Portuguese Conference on Artificial Intelligence, EPIA 2001
Porto, Portugal, December 2001
Proceedings

EPIA 01

Springer

Springer
Berlin
Heidelberg
New York
Barcelona
Hong Kong
London
Milan
Paris
Tokyo

Pavel Brazdil   Alípio Jorge (Eds.)

Progress in Artificial Intelligence

Knowledge Extraction, Multi-agent Systems,
Logic Programming, and Constraint Solving

10th Portuguese Conference on Artificial Intelligence, EPIA 2001
Porto, Portugal, December 17-20, 2001
Proceedings

Springer

Series Editors

Jaime G. Carbonell,Carnegie Mellon University, Pittsburgh, PA, USA
Jörg Siekmann, University of Saarland, Saarbrücken, Germany

Volume Editors

Pavel Brazdil
Alípio Jorge
University of Porto, Faculty of Economics
LIACC, Laboratório de Inteligência Artificial e Ciência de Computadores
Rua do Campo Alegre, 823, 4150-180 Porto, Portugal
E-mail:{pbrazdil/amjorge}@liacc.up.pt

Cataloging-in-Publication Data applied for

Die Deutsche Bibliothek - CIP-Einheitsaufnahme

Progress in artificial intelligence : knowledge extraction, multi-agent
systems, logic programming and constraint solving ; proceedings /
10th Portuguese Conference on Artificial Intelligence, EPIA 2001, Porto,
Portugal, December 17 - 20, 2001. Pavel Brazdil ; Alípio Jorge (ed.). -
Berlin ; Heidelberg ; New York ; Barcelona ; Hong Kong ; London ; Milan ;
Paris ; Tokyo : Springer, 2001
(Lecture notes in computer science ; Vol. 2258 : Lecture notes in
artificial intelligence)
ISBN 3-540-43030-X

CR Subject Classification (1998): I.2

ISBN 3-540-43030-X Springer-Verlag Berlin Heidelberg New York

Typesetting: Camera-ready by author
Printed on acid-free paper SPIN 10846042 06/3142 5 4 3 2 1 0

Preface

The tenth Portuguese Conference on Artificial Intelligence, EPIA 2001 was held in Porto and continued the tradition of previous conferences in the series. It returned to the city in which the first conference took place, about 15 years ago. The conference was organized, as usual, under the auspices of the Portuguese Association for Artificial Intelligence (APPIA, `http://www.appia.pt`). EPIA maintained its international character and continued to provide a forum for presenting and discussing researc h on different aspects of Artificial Intelligence.
To promote motivated discussions among participants, this conference strengthened the role of the thematic workshops. These were not just satellite events, but rather formed an integral part of the conference, with joint sessions when justified. This had the advantage that the work was presented to a motivated audience. This was the first time that EPIA embarked on this experience and so provided us with additional challenges.
A word of appreciation must be given to those who actively promoted and organized each of the thematic workshops:

– Fernando Moura Pires and Gabriela Guimarães, for organizing the workshop on Extraction of Knowledge from Data Bases (EKDB 2001);
– Eugénio Oliveira, for organizing the workshop on Multi Agent Systems: Theory and Applications (MASTA 2001);
– José Alferes and Salvador Abreu, for organizing the workshop on Logic Programming for Artificial Intelligence and Information Systems (LPAI);
– Pedro Barahona, for organizing the workshop on Constraint Satisfaction and Operational Research Techniques for Problem Solving (CSOR);
– Luís Torgo, for organizing the workshop on Artificial Intelligence Techniques for Financial Time Series Analysis (AIFTSA);

The organization of this volume reflects the thematic threads described above with some exceptions.
We would also like to thank all the authors who submitted the 88 articles to the conference. From these, 21 (24%) were selected as long papers for this volume. A further 18 (about 20%) were accepted as short papers.
Regarding the geographical origin, the first authors of submitted papers come from Portugal (32), Spain (17), Brazil (7), France (5), Slovenia (4), Korea (4), USA (3), Germany (3), UK (3), Sweden (3), Austria (2), and also Chile, The Netherlands, Turkey, Australia, and Poland with one submission each.
In terms of scientific areas, there were 27 submissions on knowledge extraction (10 accepted for this volume), 16 on multi-agents (9 accepted), 12 on logic programming (6 accepted), 10 on constraint solving (3 accepted), 8 on financial time series (2 accepted), and 15 on other areas (7 accepted).
Some articles not included in this volume were selected for oral presentation at the conference and/or at the thematic workshops. More information can be

found at the conference site (http://www.niaad.liacc.up.pt/Events/EPIA01/).
Special thanks to all members of the Program Committee who took upon most of the burden of reviewing. Thanks also to all other reviewers for their commitment and collaboration.
We would also like to gratefully acknowledge the support of the Portuguese Government through FCT (Fundação da Ciência e Tecnologia), University of Porto, Faculty of Economics of Porto, LIACC (Laboratório de Inteligência Artificial e Ciência de Computadores), the company Enabler, and other sponsors whose support is acknowledged at the conference site.
Finally, we thank Luís Paulo Reis for organizing the activities related to robosoccer, Rui Camacho, our Publicity Chair, Rodolfo Matos for technical support and local organization in collaboration with Rui Leite, Vera Costa, Filipa Ribeiro and Joel Silva.

October 2001 Pavel Brazdil and Alípio Jorge

EPIA'01

10th Portuguese Conference on Artificial Intelligence
Porto, Portugal, December 2001

Programme Co-Chairs

Pavel Brazdil	Univ. Porto
Alípio Jorge	Univ. Porto

Programme Commitee

Paulo Azevedo	*Univ. Minho*
Pedro Barahona	*Univ. Nova de Lisboa*
Luís Camarinha Matos	*Univ. Nova de Lisboa*
Hélder Coelho	*Univ. Lisboa*
Ernesto Costa	*Univ. Coimbra*
João Gama	*Univ. Porto*
Claude Kirchner	*LORIA-INRIA, France*
Carolina Monard	*Univ. S. Paulo, Brazil*
Fernando Moura Pires	*Univ. Évora*
Arlindo Oliveira	*IST, Lisboa*
Eugénio Oliveira	*Univ. Porto*
Ramon Otero	*Univ. Coruña, Spain*
David Pearce	*European Commission (EU)*
Gabriel Pereira Lopes	*Univ. Nova de Lisboa*
Ruy de Queiroz	*Univ. F. Pernambuco, Brazil*
Jan Rauch	*Univ. of Economics, Czech Rep.*
João Sentieiro	*IST. Lisboa*
Carles Sierra	*Univ. Catalunia, Spain*
Luís Torgo	*Univ. Porto*

List of Reviewers

Salvador Abreu
Alexandre Agustini
José Alferes
Luís Antunes
Paulo Azevedo
António L. Bajuelos
Pedro Barahona
Orlando Belo
Mário R. Benevides
Carlos L. Bento
Olivier Boissier
Rachel Bourne
Olivier Bournez
Andrea Bracciali
Pavel Brazdil
M. Paula Brito
Sabine Broda
Antoni Brogi
Francisco Bueno
Michele Bugliesi
Daniel Cabeza
Rui Camacho
Margarida Cardoso
Gladys Castillo
Luis F. Castro
Hélder Coelho
Luís Correia
Manuel E. Correia
Joaquim P. Costa
Vitor S. Costa
Carlos V. Damásio
Gautam Das
Gael H. Dias
Frank Dignum
Virginea Dignum
Jurgen Dix
Agostino Dovier
Jesus C. Fernandez
José L. Ferreira
Marcelo Finger
Miguel Filgueiras
José M. Fonseca
Dalila Fontes
Michael Fink
Michael Fisher
Ana Fred
João Gama
Pablo Gamallo
Gabriela Guimarães
Nick Jennings
Alípio Jorge
Claude Kirchner
Jörg Keller
Norbert Kuhn
King-Ip Lin
Vitor Lobo
Alneu A. Lopes
Luís S. Lopes
Gabriel P. Lopes
Inês Lynce
Luis M. Macedo
Benedita Malheiro
Margarida Mamede
Nuno C. Marques
João Marques-Silva
João P. Martins
Luis C. Matos
Carolina Monard
Nelma Moreira
Fernando Moura-Pires
João Moura Pires
Gholamreza Nakhaeizadeh
Susana Nascimento
José Neves
Vitor Nogueira
Eugénio Oliveira
Ana Paiva
João P. Pedroso
Francisco C. Pereira
Luís M. Pereira
Bernhard Pfahringer
Alessandra di Pierro
Jorge S. Pinto
Foster Provost
Paulo Quaresma
Elisa Quintarelli
Luis P. Reis
Solange O. Rezende
Ana P. Rocha
António J. Rodrigues
Irene Rodrigues
Sabina Rossi
Diana Santos
Sergey Semenov
Carles Sierra
Jaime Sichman
João M. Silva
Carlos Soares
Terrance Swift
Paulo Teles
F. P. Terpstra
Ana P. Tomás
Luís Torgo
Tarma Uustalu
Vasco T. Vasconcelos
Gerard Widmer
Marijana Zekic-Susac
Jan Žizka

Table of Contents

Agent Programming in Ciao Prolog

Francisco Bueno and the CLIP Group[⋆]

Facultad de Informática – UPM
bueno@fi.upm.es

The agent programming landscape has been revealed as a natural framework for developing "intelligence" in AI. This can be seen from the extensive use of the agent concept in presenting (and developing) AI systems, the proliferation of agent theories, and the evolution of concepts such as agent societies (social intelligence) and coordination.

Although a definition of what is an agent might still be controversial, agents have particular characteristics that define them, and are commonly accepted. An agent has autonomy, reactivity (to the environment and to other agents), intelligence (i.e., reasoning abilities). It behaves as an individual, capable of communicating, and capable of modifying its knowledge and its reasoning.

For programming purposes, and in particular for AI programming, one would need a programming language/system that allows to reflect the nature of agents in the code: to map code to some abstract entities (the "agents"), to declare well-defined interfaces between such entities, their individual execution, possibly concurrent, possibly distributed, and their synchronization, and, last but not least, to program intelligence.

It is our thesis that for the last purpose above the best suited languages are logic programming languages. It is arguably more difficult (and unnatural) to incorporate reasoning capabilities into, for example, an object oriented language than to incorporate the other capabilities mentioned above into a logic language. Our aim is, thus, to do the latter: to offer a logic language that provides the features required to program (intelligent) agents comfortably.

The purpose of this talk, then, is not to introduce sophisticated reasoning theories or coordination languages, but to go through the (low-level, if you want) features which, in our view, provide for agent programming into a (high-level) language, based on logic, which naturally offers the capability of programming reasoning.

The language we present is Ciao, and its relevant features are outlined below. Most of them have been included as language-level extensions, thanks to the extensibility of Ciao. Hopefully, the Ciao approach will demonstrate how the required features can be embedded in a logic programming language in a natural way, both for the implementor and for the programmer.

State and Its Encapsulation: From Modules to Objects The state that is most relevant for programming intelligence is the state of knowledge. Classically, a logic program models a knowledge state, and, also, logic languages provide the means

[⋆] Daniel Cabeza, Manuel Carro, Jesús Correas, José M. Gómez, Manuel Hermenegildo, Pedro López, Germán Puebla, and Claudio Vaucheret to manipulate and change this state: the well-known assert/retract operations. These can be used for modeling knowledge evolution. On the other hand, state of data and its evolution are arguably best modeled with objects. Also, objects are a basic construct for capturing "individuality" in the sense of agents.

What is needed is a neat differentiation between the code that represents the state of knowledge and the code that represents the evolution of knowledge. A first step towards this is providing state encapsulation. This can be achieved with a well-defined, well-behaved module system. This is one of the main principles that has informed the design of Ciao. Having a language with modules and encapsulated state, the step towards objects is an easy one: the only extra thing needed is instantiation. Once we add the ability to create instances of modules, we have classes. This has been the approach in O'Ciao, the Ciao sublanguage for object orientation.

Concurrency and Distribution These two features are provided in Ciao at two levels. At the language level, there are constructs for concurrent execution and for distributed execution. At the level of "individual entities", concurrency and distribution comes through via the concept of active modules/objects.

Reactivity and Autonomy: Active Modules/Objects A module/object is active when it can run as a separate process. This concept provides for "autonomy" at the execution level. The active module service of Ciao allows starting and/or connecting (remote) processes which "serve" a module/object. The code served can be conceptually part of the application program, or it can be viewed alternatively as a program *component*: a completely autonomous, independent functionality, which is given by its interface.

Active modules/objects can then be used as "watchers" and "actors" of and on the outside world. The natural way to do this is from a well-defined, easy-to-use foreign language interface that allows to write drivers that interact with the environment, but which can be viewed as logic facts (or rules) from the programmer's point of view. An example of this is the SQL interface of Ciao to sql-based external databases.

And More Adding more capabilities to the language, in particular, adding more sophisticated reasoning schemes, requires that the language be easily extensible. Extensibility has been another of the guidelines informing the design of Ciao: the concept of a package is a good example of this. Packages are libraries that allow syntactic, and also semantic, extensions of the language. They have been already used in Ciao, among other things (including most of the abovementioned features), to provide higher order constructions like predicate abstractions, and also fuzzy reasoning abilities.

Multi-relational Data Mining: A Perspective

Peter A. Flach

Department of Computer Science, University of Bristol
Woodland Road, Bristol BS8 1UB, United Kingdom
`Peter.Flach@bristol.ac.uk`, `www.cs.bris.ac.uk/~flach/`

Abstract. Multi-relational data mining (MRDM) is a form of data mining operating on data stored in multiple database tables. While machine learning and data mining are traditionally concerned with learning from single tables, MRDM is required in domains where the data are highly structured. One approach to MRDM is to use a predicate-logical language like clausal logic or Prolog to represent and reason about structured objects, an approach which came to be known as inductive logic programming (ILP) [18,19,15,16,13,17,2,5].
In this talk I will review recent developments that have led from ILP to the broader field of MRDM. Briefly, these developments include the following:

– the use of other declarative languages, including functional and higher-order languages, to represent data and learned knowledge [9,6,1];
– a better understanding of knowledge representation issues, and the importance of data modelling in MRDM tasks [7,11];
– a better understanding of the relation between MRDM and standard single-table learning, and how to upgrade single-table methods to MRDM or downgrade MRDM tasks to single-table ones (propositionalisation) [3,12,10,14];
– the study of non-classificatory learning tasks, such as subgroup discovery and multi-relational association rule mining [8,4,21];
– the incorporation of ROC analysis and cost-sensitive classification [20].

References

1. A. F. Bowers, C. Giraud-Carrier, and J. W. Lloyd. Classification of individuals with complex structure. In *Proceedings of the 17th International Conference on Machine Learning*, pages 81–88. Morgan Kaufmann, June 2000.
2. L. De Raedt, editor. *Advances in Inductive Logic Programming*. IOS Press, 1996.
3. L. De Raedt. Attribute value learning versus inductive logic programming: The missing links (extended abstract). In D. Page, editor, *Proceedings of the 8th International Conference on Inductive Logic Programming*, volume 1446 of *Lecture Notes in Artificial Intelligence*, pages 1–8. Springer-Verlag, 1998.
4. Luc Dehaspe and Hannu Toivonen. Discovery of relational association rules. In Saso Dzeroski and Nada Lavrac, editors, *Relational Data Mining*, pages 189–212. Springer-Verlag, September 2001.
5. Saso Dzeroski and Nada Lavrac, editors. *Relational Data Mining*. Springer-Verlag, Berlin, September 2001.
6. P.A. Flach, C. Giraud-Carrier, and J.W. Lloyd. Strongly typed inductive concept learning. In D. Page, editor, *Proceedings of the 8th International Conference on Inductive Logic Programming*, volume 1446 of *Lecture Notes in Artificial Intelligence*, pages 185–194. Springer-Verlag, 1998.

7. Peter A. Flach. Knowledge representation for inductive learning. In Anthony Hunter and Simon Parsons, editors, *Symbolic and Quantitative Approaches to Reasoning and Uncertainty (ECSQARU'99)*, volume 1638 of *Lecture Notes in Artificial Intelligence*, pages 160–167. Springer-Verlag, July 1999.
8. Peter A. Flach and Nicolas Lachiche. Confirmation-guided discovery of first-order rules with tertius. *Machine Learning*, 42(1/2):61–95, January 2001.
9. J. Hernández-Orallo and M.J. Ramírez-Quintana. A strong complete schema for inductive functional logic programming. In S. Džeroski and P. Flach, editors, *Proceedings of the 9th International Workshop on Inductive Logic Programming*, volume 1634 of *Lecture Notes in Artificial Intelligence*, pages 116–127. Springer-Verlag, 1999.
10. Stefan Kramer, Nada Lavrac, and Peter Flach. Propositionalization approaches to relational data mining. In Saso Dzeroski and Nada Lavrac, editors, *Relational Data Mining*, pages 262–291. Springer-Verlag, September 2001.
11. Nicolas Lachiche and Peter Flach. A first-order representation for knowledge discovery and bayesian classification on relational data. In Pavel Brazdil and Alipio Jorge, editors, *PKDD2000 workshop on Data Mining, Decision Support, Meta-learning and ILP : Forum for Practical Problem Presentation and Prospective Solutions*, pages 49–60. 4th International Conference on Principles of Data Mining and Knowledge Discovery, September 2000.
12. Wim Van Laer and Luc De Raedt. How to upgrade propositional learners to first order logic: A case study. In Saso Dzeroski and Nada Lavrac, editors, *Relational Data Mining*, pages 235–261. Springer-Verlag, September 2001.
13. N. Lavrač and S. Džeroski. *Inductive Logic Programming: Techniques and Applications*. Ellis Horwood, 1994.
14. Nada Lavrač and Peter A. Flach. An extended transformation approach to inductive logic programming. *ACM Transactions on Computational Logic*, 2(4):458–494, October 2001.
15. S. Muggleton. Inductive logic programming. *New Generation Computing*, 8(4):295–317, 1991.
16. S. Muggleton, editor. *Inductive Logic Programming*. Academic Press, 1992.
17. S. Muggleton and L. De Raedt. Inductive logic programming: Theory and methods. *Journal of Logic Programming*, 19/20:629–679, 1994.
18. G.D. Plotkin. *Automatic Methods of Inductive Inference*. PhD thesis, Edinburgh University, 1971.
19. E.Y. Shapiro. *Algorithmic Program Debugging*. The MIT Press, 1983.
20. Ashwin Srinivasan. Context-sensitive models in inductive logic programming. *Machine Learning*, 43(3):301–324, September 2001.
21. Stefan Wrobel. Inductive logic programming for knowledge discovery in databases. In Saso Dzeroski and Nada Lavrac, editors, *Relational Data Mining*, pages 74–101. Springer-Verlag, September 2001.

A Comparison of GLOWER and Other Machine Learning Methods for Investment Decision Making

Vasant Dhar

Stern School of Business, New York University
44 West 4th Street, Room 9-75, New York NY 10012 vdhar@stern.nyu.edu

Abstract. Prediction in financial domains is notoriously difficult for a number of reasons. First, theories tend to be weak or non-existent, which makes problem formulation open ended by forcing us to consider a large number of independent variables and thereby increasing the dimensionality of the search space. Second, the weak relationships among variables tend to be nonlinear, and may hold only in limited areas of the search space. Third, in financial practice, where analysts conduct extensive manual analysis of historically well performing indicators, a key is to find the hidden interactions among variables that perform well in combination. Unfortunately, these are exactly the patterns that the greedy search biases incorporated by many standard rule learning algorithms will miss.
One of the basic choices faced by modelers is on the choice of search method to use. Some methods, notably, tree induction provide explicit models that are easy to understand. This is a big advantage of such methods over, say, neural nets or naïve Bayes. My experience in financial domains is that decision makers are more likely to invest capital using models that are easy to understand. More specifically, decision makers want to understand when to pay attention to specific market indicators, and in particular, in what ranges and under what conditions these indicators produce good risk- adjusted returns. Indeed, many professional traders have remarked that they are occasionally inclined to make predictions about market volatility and direction, but cannot specify these conditions precisely or with any degree of confidence. For this reason, rules generated by pattern discovery algorithms are particularly appealing in this respect because they can make explicit to the decision maker the particular interactions among the various market indicators that produce desirable results. They can offer the decision maker a "loose theory" about the problem that is easy to critique.
In this talk, I describe and evaluate several variations of a new genetic learning algorithm (GLOWER) on a variety of data sets. The design of GLOWER has been motivated by financial prediction problems, but incorporates successful ideas from tree induction and rule learning. I examine the performance of several GLOWER variants on a standard financial prediction problem (S&P500 stock returns), using the results to identify one of the better variants for further comparisons. I introduce a new (to KDD) financial prediction problem (predicting positive and negative earnings surprises), and experiment with GLOWER, contrasting it with tree- and rule-induction approaches as well as other approaches such as neural nets and naïve Bayes. The results are encouraging, showing that GLOWER has the ability to uncover effective patterns for difficult problems that have weak structure and significant nonlinearities. Finally, I shall discuss open issues such as the difficulties of dealing with non stationarity in financial markets.

Parallel Implementation of Decision Tree Learning Algorithms

Nuno Amado, João Gama, and Fernando Silva

LIACC - University of Porto
R.Campo Alegre 823, 4150 Porto
{namado,jgama,fds}@ncc.up.pt

Abstract. In the fields of data mining and machine learning the amount of data available for building classifiers is growing very fast. Therefore, there is a great need for algorithms that are capable of building classifiers from very-large datasets and, simultaneously, being computationally efficient and scalable. One possible solution is to employ parallelism to reduce the amount of time spent in building classifiers from very-large datasets and keeping the classification accuracy. This work first overviews some strategies for implementing decision tree construction algorithms in parallel based on techniques such as task parallelism, data parallelism and hybrid parallelism. We then describe a new parallel implementation of the C4.5 decision tree construction algorithm. Even though the implementation of the algorithm is still in final development phase, we present some experimental results that can be used to predict the expected behavior of the algorithm.

1 Introduction

Classification has been identified as an important problem in the areas of data mining and machine learning. Over the years different models for classification have been proposed, such as neural networks, statistical models as linear and quadratic discriminants, decision trees, and genetic algorithms. Among these models, decision trees are particularly suited for data mining and machine learning. Decision trees are relatively faster to build and obtain similar, sometimes higher, accuracy when compared with other classification methods [7,9]. Nowadays there is an exponential growing on the data stored in computers. Therefore, it is important to have classification algorithms computationally efficient and scalable. Parallelism may be a solution to reduce the amount of time spent in building decision trees using larger datasets and keep the classification accuracy [4,5,11,12]. Parallelism can be easily achieved by building the tree decision nodes in parallel or by distributing the training data. However, implementing parallel algorithms for building decision trees is a complex task due to the following reasons. First, the shape of the tree is highly irregular and it is determined only at runtime, beyond the fact that the amount of processing used for each node varies. Hence, any static allocation scheme will probably suffer from a high load imbalance problem. Second, even if the successors of a node are processed in parallel, their construction needs part of the data associated to the parent. If the data is dynamically distributed by the processors witch are responsible for different nodes then it is necessary to implement data movements. If the data is not properly distributed then the performance of the algorithm can suffer due to a loss of locality [5,12]. Decision trees are usually built in two phases: tree construction and simplification phases. With no doubt, the most computational expensive phase of the algorithm is the tree construction phase[2,9,10,12]. Therefore, in this work, for the description of the parallel construction tree algorithms we only consider the first phase.

In the reminder of the paper we first overview some strategies for implementing parallel decision tree algorithms, and then describe our parallel implementation of the C4.5 decision tree construction algorithm. We then present some preliminary results and draw some conclusions.

2 Related Work

This section overviews some strategies for implementing decision tree construction algorithms in parallel using task parallelism, data parallelism and hybrid parallelism.

Task parallelism. The construction of decision trees in parallel by following a task-parallelism approach can be viewed as dynamically distributing the decision nodes among the processors for further expansion. A single processor using all the training set starts the construction phase. When the number of decision nodes equals the number of processors the nodes are split among them. At this point each processor proceeds with the construction of the decision sub-trees rooted at the nodes of its assignment. This approach suffers, in general, from bad load balancing due to the possible different sizes of the trees constructed by each processor. Also, for the implementations presented in [4], they require the whole training set to be replicated in the memory of all the processors. Alternatively, they require a great amount of communications for each processor to have access to the examples kept in the other's memory.

Data parallelism. The use of data parallelism in the design of parallel decision tree construction algorithms can be generally described as the execution of the same set of instructions (algorithm) by all processors involved. The parallelism is achieved by distributing the training set among the processors where each processor is responsible for a distinct set of examples. The distribution of the data can be performed in two different ways, horizontally or vertically.

The parallel strategy based on vertical data distribution[5] consists in splitting the data by assigning a distinct set of attributes to each processor. Each processor keeps in its memory only the whole values for the set of attributes assigned to him and the values of the classes. During the evaluation of the possible splits each processor is responsible only for the evaluation of its attributes. Parallelism with vertical data distribution can still suffer from load imbalance due to the evaluation of continuous attributes which requires more processing than the evaluation of discrete attributes.

The parallel strategy based on horizontal data distribution consists in distributing the examples evenly by the processors. Each processor keeps in its memory only a distinct subset of examples of the training set. The possible splits of the examples associated to a node are evaluated by all the processors, which communicate between them to find the global values of the criteria used and, by this, the best split. Each processor performs locally the split. In [11], the authors describe the implementation of two algorithms with horizontal data distribution: SLIQ and SPRINT systems. For the evaluation of a split based on a continuous attribute, these systems have the set of examples sorted by the values of the attribute. In order to avoid sorting the examples every time a continuous attribute is evaluated, they use separate lists of values for each attribute, which are sorted once at the beginning of the tree construction. SLIQ also uses a special list, called the class list, which has the values of the class for each example and stays resident in memory during all the tree construction process. In the parallel implementation of SLIQ the training set is distributed horizontally among the processors where each processor is responsible for creating its own attribute lists. In SPRINT, the class list is eliminated by adding the class label in the attribute lists entries. The index in the attribute lists is now the index of the example in the training set. The evaluation of the possible splits is also performed by all processors, which communicate between them to find the best split. After finding the best split, each processor is responsible for splitting its own attribute lists. Both systems report good performance and scalability results, being also capable of processing very-large datasets. However, the operation to perform the split of the set of examples associated with a node requires high communication load in both systems.

Hybrid parallelism. The parallel decision tree construction algorithms, which use hybrid parallelism, can be characterized as using both data parallelism with horizontal or vertical distribution and task parallelism. The implementation of hybrid parallel algorithms is strongly motivated by the choice between the distribution of the amount of processing at each node and the required volume of communications. For the nodes covering a significant amount of examples, is used data parallelism to avoid the problems already stated of load imbalance and of poor use of parallelism associated with task parallelism. But, for the nodes covering fewer examples the time used for communications can be higher than the time spent in processing the examples. To avoid this problem, when the number of examples associated to a node is lower than a specific value, one of the processors continues alone the construction of the tree rooted at the node (task parallelism). Two parallel decision tree construction algorithms using hybrid parallelism are described in [6,12].

3 C4.5 Parallel Implementation

In this section, we describe a new parallel implementation of the C4.5 decision tree construction algorithm. This implementation follows an horizontal data parallelism strategy similar to that used by the parallel implementation of SLIQ [11], but modified for the C4.5 algorithm. Our parallel implementation was designed to be executed in a distributed memory environment where each of the k processors has its own private memory. It addresses two fundamental issues: load balance and locality. An uniform load balance is achieved by distributing horizontally the examples equally among the k processors and using a new breadth-first strategy for constructing the decision tree. As in the parallel version of SLIQ, each processor is responsible for building its own attribute lists and class lists from the subset of examples assigned to it. The entries in the class lists keep, for each example, the class label, the weight (used in the C4.5 algorithm for dealing with unknown values), the corresponding global index of the example in the training set and a pointer to the node in the tree to which the example belongs. The attribute lists also have an entry for each example with the attribute value and an index pointing to the example corresponding entry in the class list. The continuous attribute lists are globally sorted by the values of the attribute using the sorting algorithm described in [3]. Each processor has now, for each continuous attribute, a list of sorted values where the first processor has the lower values of the attribute, the second has the attribute values higher then the first processor and lower then the third, and so on. Because of the global sort of the continuous attributes, a processor can have attribute values that do not correspond to any of the examples initially assigned to it. After the sort, each processor updates its class list with the examples information corresponding to the new values of the continuous attributes.

Several authors[2,10] pointed out one of the efficiency limitations of C4.5: the repetitive sorting of the examples covered by a node every time a continuous attribute is evaluated. This limitation is eliminated with the implementation of the attribute lists where the continuous attributes are sorted only once. The main problems in the parallel tree construction process reside in performing the split and in finding the best split for the set of examples covered by a node. For these two steps, communication among processors is necessary in order to determine the best global split and the assignments of examples to the new subsets resulting from the split.

In the parallel version of the algorithm each processor has a distinct subset of values of the continuous attributes in their globally sorted lists. Before each processor starts evaluating the possible splits of the examples, the distributions must be initialized to reflect the examples assigned to the other processors. Each processor finds its distributions of the local set of examples for each attribute and sends them to the other processors. For the nominal attributes, these distributions are gathered in all processors. All processors compute the gain for nominal attributes. For continuous attributes the gain of a possible split, is found based upon the distributions before and after the split point. When a processor receives the distributions from another it initializes, for each continuous attribute, the distributions before the split point with the distributions of the processors with lower rank. The distributions after the split point are initialized with those from the processors with higher rank. After evaluating all possible divisions of their local set of examples, the processors communicate among themselves in order to find the best split from the k local best splits found by each one.

As soon as the best split is found, the split is performed by creating the child nodes and dividing the set of examples covered by the node. This step requires for each processor to update the pointers to the nodes in their class lists. The attribute lists are divided in many lists, as a result of the split, and each of those sublists are assigned to every new child node. For the split each processor scans its local list of the attribute chosen and depending on the value of the attribute, each entry is moved to the correspondent sublist, and the pointer to the node in the class list is updated.

For dividing the remaining attribute lists the pointers to the nodes in the class list are used. But, before that, the class lists in each process must be updated. For the entries in the class list of one processor, whose values for the chosen attribute are kept in another processor, must be updated with the information of the node to which the corresponding examples were assigned by the other processor. For each example in the set covered by the node, the index, weight and the node to which they were assigned are gathered in all processors allowing each processor to update its local class list and divide the attribute lists. When dividing the attribute lists, each processor finds its local class distributions for each attribute in the new sets of examples assigned to the child nodes. Still on this phase, the distributions are scattered allowing for each processor to know the global class distributions used later during the splits evaluation phase.

Our parallel decision tree algorithm described in this work preserves most of the main features of C4.5. One of the most important is the ability to deal with unknown attribute values. Furthermore, since the same evaluation criteria and splits were considered in the parallel version, it obtains the same decision trees and classification results as those obtained by C4.5. Our parallel system modifies the C4.5 algorithm in order to use attribute lists and class lists. These are fundamental data structures to achieve parallelism as used in SLIQ strategy. The main reason for using separate lists for the attributes is to avoid sorting the examples every time a continuous attribute is evaluated. The several sorts performed by C4.5 during the evaluation of the attributes are one of its efficiency limitations [2,10], and, if they were kept in the parallel version then they would strongly limit the performance of the algorithm. Hence, by using separate lists for the attributes makes it possible to globally sort the continuous attributes only once at the beginning of the tree construction process.

The SPRINT strategy was developed to overcome the use of centralized structures such as the class list. It avoids the class lists by extending the attribute lists with two extra fields, the class label and the global index of the example in the training set. The main disadvantage stated in [11] for the use of the class list is that it can limit the size of the training set that can be used with the algorithm, due to the fact that it must stay resident in memory during the execution of the algorithm. Suppose now we have a training set with N examples and A attributes and the entry values of the attribute lists and class list require one word of memory. Suppose also that we have k processors available for the execution of the algorithms. In this situation, SPRINT will need $3 * A * N/k$ memory words in each processor to store the training set divided equally among the k processors. For the algorithm described in this work the worst situation happens when, due to the global sort of the continuous attribute lists, the class list must store the information for all examples in the training set. In this situation the amount of memory necessary to store the training set in each processor will be $2 * A * N/k + 3 * N$ memory words. It is easy to observe that when the number of attributes is three times greater then the number of available processors ($A > 3*k$) it is better to use, in terms of memory, the class list then the attribute lists. Another disadvantaged stated in [11] relatively to the use of the class list is the time required to update it during the split. In the implementation of SLIQ, the class list is entirely replicated in the memory of each processor. After the split each processor has to update the entire list, which limits the performance of the algorithm.

Similarly to SLIQ and SPRINT, in the parallel version of C4.5 for the split the processors exchange the global indexes of the examples covered by a node with the information to which child node they were assigned. Each processor keeps a list of pointers to the entries of the class list sorted by the examples indexes in order to improve the searches when updating the class list with the information gathered from the other processors.

4 Experimental Results

Our parallel implementation is still being refined, therefore the results are only preliminary and we believe are subject to further improvement. However, they can be used to predict the expected behavior of the parallel algorithm. As mentioned before, our decision tree algorithm uses the same criteria and procedures used in the C4.5 algorithm. Therefore, they produce the same decision trees with equal classification accuracy. Consequently, we evaluate the performance of our system only in terms of execution time required for the construction of the decision tree. Again, the prune phase of the algorithm was not taken into account for the time measured in the experiments as it is negligible.

The parallel version of C4.5 was implemented using the standard MPI communication primitives [8]. The use of standard MPI primitives allows the implementation of the algorithm to be highly portable to clusters of machines, that is to distributed memory environments. The experiments presented here were conducted on a PC server with four Pentium Pro CPUs and 256MB of memory running Linux 2.2.12 from the standard RedHat Linux release 6.0 Distribution. Each CPU in the PC server runs at 200MHz and contains 256Kb of cache memory. All the time results presented here are the average result of three executions of the algorithm.

All experiments used the Synthetic data set from Agrawal et. al. [1]. This dataset has been previously used for testing parallel decision tree construction algorithms such as the systems SLIQ and SPRINT [11,12]. Each example in the dataset has 9 attributes, where 5 of them are continuous and 3 are discrete. The

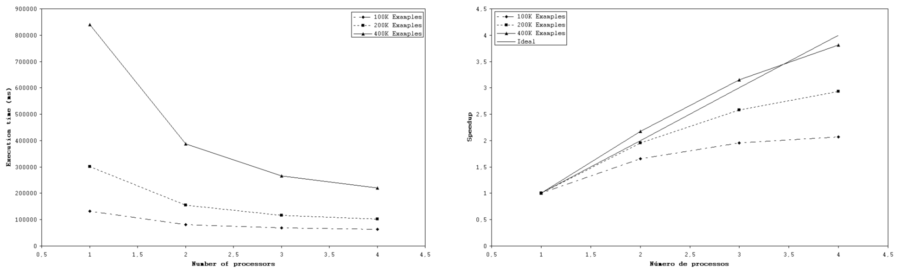


Fig. 1. Speedup results.

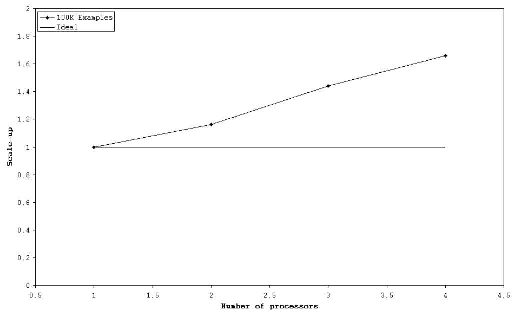


Fig. 2. Scale-up results.

values of the attributes are randomly generated. Agrawal et. al. also describe 10 classification functions based on the values of these attributes and with different complexities. For the datasets used in the experiments it was only considered one of those functions, named *Function2*.

The first set of experiments measured the performance, in terms of speedup, of the algorithm. For each experiment we kept constant the total size of the training set and varying the number of processors from 1 to 4. Figure 1 illustrates the speedup results for training sets of 100k, 200k and 400k examples. Our algorithm overall shows good speedup results and, as expected, the best results were obtained for the largest dataset used (with 400K examples). The slowdown observed for the largest processor configuration is mainly due to the high communication costs in the split phase. The next set of experiments aimed to test the scale-up characteristics of the algorithm. In these experiments, for each scale-up measure, the size of the training set (100k examples) in each processor was kept constant while the processors configuration varied from 1 to 4. Figure 2 shows that our algorithm achieves good scale-up behavior. Again, high communication overheads in the splitting phase influence scale-up capacity as the processors increase.

5 Conclusion

In this work we overviewed some parallel algorithms for constructing decision trees and proposed a new parallel implementation of the C4.5 decision tree construction algorithm. Our parallel algorithm uses a strategy similar to that of SLIQ but adapted for the C4.5 algorithm. Furthermore, it preserves the main features of C4.5, such as the ability to deal with unknown attribute values. As a result, our system builds the same decision trees, as those obtained by C4.5, with the same classification accuracy.

Based in the preliminary results presented, the algorithm shows good speedup and scale-up performance indicating that it can be used to build decision trees from very-large datasets. However some work has yet to be done. The results obtained when measuring the speedup characteristics of the algorithm indicate that the communications overhead is a key factor in limiting the performance of the algorithm.

Acknowledgements

This work was partially supported by funds granted to LIACC through the Programa de Financiamento Plurianual, Fundação de Ciência e Tecnologia, Programa POSI and FEDER. J.Gama acknowledges Metal and Sol-Eu-Net projects.

References

1. R. Agrawal, T. Imielinski, and A. Swami. Database mining: A performance perspective. In *Special Issue on Learning and Discovery in Knowledge-Based Databases*, number 5(6), pages 914–925. IEEE, 1993.
2. J. Catlett. *Megainduction: Machine learning on very large databases.* PhD thesis, University of Sydney, 1991.
3. D. J. DeWitt, J. F. Naughton, and D. A. Schneigder. Parallel sorting on a shared-nothing architecture using probabilistic splitting. In *Proc. of the 1st Int'l Conf. on Parallel and Distributed Information Systems*, 1991.
4. J. Darlington et. al. Parallel induction algorithms for data mining. In *Proc. of 2nd Intelligent Data Analysis Conference.* Springer Verlag, LNCS 1280, 1997.
5. A. A. Freitas and S. H. Lavington. *Mining Very Large Databases with Parallel Processing.* Kluwer Academic Publishers, 1998.
6. R. Kufrin. Decision trees on parallel processors. *Parallel Processing for Artificial Intelligence, Elsevier*, 3:279–306, 1997.
7. D. Michie, D. J. Spiegelhalter, and C. C. Taylor. *Machine Learning, Neural and Statistical Classification.* Ellis Horwood, 1994.
8. P. S. Pacheco. *Parallel Programming with MPI.* Morgan Kaufmann Pubs., 1997.
9. J. R. Quinlan. *C4.5: Programs for Machine Learning.* Morgan Kaufmann Publishers, Inc., 1992.
10. S. Ruggieri. Efficient c4.5. Technical report, Universit di Pisa, 2000.
11. J. C. Shafer, R. Agrawal, and M. Mehta. SPRINT: A scalable parallel classifier for data mining. In *Proc. 22nd Int. Conf. Very Large Databases*, pages 544–555. Morgan Kaufmann, 3–6 1996.
12. A. Srivastava, E. Han, V. Kumar, and V. Singh. Parallel formulations of decision-tree classification algorithms. *Data Mining and Knowledge Discovery*, 3, 1999.

Reducing Rankings of Classifiers by Eliminating Redundant Classifiers

Pavel Brazdil[1], Carlos Soares[1], and Rui Pereira[2]

[1] LIACC / FEP, University of Porto, Rua do Campo Alegre, 823, 4150-180 Porto, Portugal, {pbrazdil,csoares}@liacc.up.pt

[2] Infopulse, Portugal, rpereira@infopulse.pt

Abstract. Several methods have been proposed to generate rankings of supervised classification algorithms based on their previous performance on other datasets [8,4]. Like any other prediction method, ranking methods will sometimes err, for instance, they may not rank the best algorithm in the first position. Often the user is willing to try more than one algorithm to increase the possibility of identifying the best one. The information provided in the ranking methods mentioned is not quite adequate for this purpose. That is, they do not identify those algorithms in the ranking that have reasonable possibility of performing best. In this paper, we describe a method for that purpose. We compare our method to the strategy of executing all algorithms and to a very simple reduction method, consisting of running the top three algorithms. In all this work we take time as well as accuracy into account. As expected, our method performs better than the simple reduction method and shows a more stable behavior than running all algorithms.

Keywords: meta-learning, ranking, algorithm selection

1 Introduction

There is a growing interest in providing methods that would assist the user in selecting appropriate classification (or other data analysis) algorithms. Some meta-learning approaches attempt to suggest one algorithm while others provide a ranking of the candidate algorithms [8,4]. The ranking is normally used to express certain expectations concerning performance (e.g. accuracy). The algorithm that appears in a certain position in the ranking is expected to be in some way better (or at least not worse) than the ones that appear afterwards. So, if the user is looking for the best performing algorithm, he can just follow the order defined by the ranking and try the corresponding algorithms out. In many cases the user may simply use the algorithm at the top of the ranking. However, the best performing algorithm may appear further on in the ranking. So, if the user is interested to invest more time in this, he can run few more algorithms and identify the algorithm that is best. Therefore our aim is (a) to come up with a ranking that includes only some of the given candidate algorithms, (b) not to decrease the chance that the best performing algorithm is excluded from the ranking. In this paper we describe an extension to the basic ranking method that has been designed for this aim.

In the next section we briefly review the basic ranking method. Section 3 describes the method used to exclude items from a given ranking. Section 4 presents the evaluation method adopted, while the following section describes the experimental setting and results. In Section 6 some alternative evaluation methods are briefly outlined and the last section presents the conclusions.

2 Constructing Rankings of Classifiers on Past Experience

Earlier we have proposed a method [8] that divides the problem of generating a ranking into two distinct phases. In the first one we employ a k-Nearest Neighbor method to identify datasets similar to the one at hand. This is done on the basis of different data characterization measures. In the second phase we consider the performance of the given classification algorithms on the datasets identified. This method is based on the Adjusted Ratio of Ratios (ARR), a multicriteria evaluation measure combining accuracy and time to assess relative performance. In ARR the compromise between the two criteria involved is given by the user in the form "the amount of accuracy I'm willing to trade for a 10 times speed-up is *AccD*%". The ARR of algorithm *ap* when compared to algorithm *aq* on data set *di* is defined as follows:

$$ARR^{di}_{ap,aq} = \frac{\frac{A^{di}_{ap}}{A^{di}_{aq}}}{1 + \log\left(\frac{T^{di}_{ap}}{T^{di}_{aq}}\right) * AccD}$$

where A^{di}_{ap} and T^{di}_{ap} are the accuracy and time of *ap* on *di*, respectively. Assuming without loss of generality that algorithm *ap* is more accurate but slower than algorithm *aq* on data set *di*, the value of $ARR^{di}_{ap,aq}$ will be 1 if *ap* is *AccD*% more accurate than *aq* for each order of magnitude that it is slower than *aq*. $ARR^{di}_{ap,aq}$ will be larger than one if the advantage in accuracy of *ap* is *AccD*% for each additional order of magnitude in execution time and vice-versa.

The individual ARR measures are used as a basis for generating a ranking. Basically, the method aggregates the information for different datasets by calculating overall means of individual algorithms:

$$ARR_{ap} = \frac{\sum_{aq} \sum_{di} ARR^{di}_{ap,aq}}{n(m-1)} \tag{1}$$

where ARR_{ap} represents the mean of individual ARR measures for algorithm *ap*, *n* represents the number of datasets and *m* the number of algorithms. The values of ARR_{ap} are then examined and the algorithms ordered according to these values.

As it has been shown this method provides quite good rankings overall and is quite competitive when compared to other approaches, such as DEA [4]. One disadvantage of this method is that it does not provide any help as to how many algorithms the user should actually try out. This problem is addressed in the next section.

3 Reducing Rankings of Classifiers by Eliminating Redundant Classifiers

There are really two reasons why we would want to eliminate algorithms from a given ranking. The first one is that the ranking recommended for a given dataset may include one or more algorithms that have proven to be significantly worse than others in similar datasets in the past and, thus, have virtually no chance of achieving better results than its competitors on the current dataset. These algorithms can thus be dropped. The second one is that the given ranking may include one or more clones of a given algorithm, that are virtually indistinguishable from it, or other algorithms with very similar performance. These algorithms can again be dropped.

We may ask why we would want to include clones in the candidate set in the first place. The answer is that recognizing clones is not an easy matter. Different people may develop different algorithms and give them different names. These, in some cases, may turn out to be quite similar in the end. There is another reason for that. We may on purpose decide to include the same algorithm several times with different parameter settings. We must thus have a way to distinguish useful variants from rather useless clones. Our objective here is to describe a method that can be used to do that.

Suppose a given ranking R consists of n classifiers, $C_1 ... C_n$. Our aim is to generate a reduced ranking R', which contains some of the items in R. The method considers all algorithms in the ranking, following the order in which they appear. Suppose, we have come to algorithm C_i. Then the sequence of subsequent algorithms C_j (where $j > i$) is then examined one by one. The aim is to determine whether each algorithm C_j should be left in the ranking, or whether it should be dropped. This decision is done on the basis of past results on datasets that are similar to the one at hand, i.e. datasets where the performance of the algorithms is similar to the performance that is expected on the new dataset. If the algorithm C_j has achieved significantly better performance than C_i on some datasets (shortly $C_j \gg C_i$) the algorithm is maintained. If it did not, then two possibilities arise. The first one is that C_j is significantly worse than C_i ($C_j \ll C_i$) on some datasets, and comparable to C_i on others. In this case C_j can be dropped. The second possibility is that neither algorithm has significant advantage over the other ($C_j \approx C_i$), indicating that they are probably clones. In this case too, C_j can be dropped. This process is then repeated until all combinations have been explored.

Here performance is judged according to the ARR measure presented earlier, which combines both accuracy and time. The historical data we use contains information on different folds of cross validation procedure. It is thus relatively easy to conduct a statistical test determining whether $ARR^{di}_{Cj,Ci}$ is consistently greater than 1 on different folds, indicating that C_j is significantly better than C_i on dataset di.

After the reduction process has terminated, the resulting ranking will contain the first item that was in R, plus a few more items from this ranking. Note that this need not be a sequence of several consecutive items. Should the best performing algorithm have a clone under a different name, this clone will be left out.

Reducing a subsequence of algorithms is beneficial in general, as it reduces the time the user needs to spend experimenting. However, this process involves certain risks. If the sequence is reduced too much, the optimal classifier may be missed out. If on the other hand the sequence is longer than required, the total time required to determine which algorithms is best increases. It is therefore necessary to have a way of evaluating different alternatives and comparing them. This topic is described in the next section.

4 Comparing Different Rankings of Uneven Length

Let us now consider the next objective - how to compare two rankings and determine which one is better. In our previous work [1,8,9] we have used a method that involves calculating a rank correlation coefficient to a ranking, constructed simply by running the algorithms on the dataset at hand.

This method has one disadvantage - it does not enable us to compare rankings of different sizes. We have therefore decided to use another method here. We evaluate a reduced ranking as if it was a single algorithm with the accuracy of the most accurate algorithm included in the ranking and with total time equal to the sum of the times of all algorithms included. Given that ARR was the performance measure used to construct and reduce the rankings, it makes sense to use the same scheme in the evaluation.

5 Experimental Evaluation

The meta-data used was obtained from the METAL project (`http://www.metal-kdd.org`). It contains results of 10 algorithms on 53 datasets. We have used three decision tree classifiers, C5.0 tree, C5.0 with boosting [7] and Ltree, which is a decision tree which can introduce oblique decision surfaces [3]. We have also used a linear discriminant (LD) [6], two neural networks from the SPSS Clementine package (Multilayer Perceptron and Radial Basis Function Network) and two rule-based systems, C5.0 rules and RIPPER [2]. Finally, we used the instance-based learner (IBL) and naive bayes (NB) implementations from the MLC++ library [5].

As for the datasets, we have used all datasets from the UCI repository with more than 1000 cases, plus the Sisyphus data and a few applications provided by DaimlerChrysler. The algorithms were executed with default parameters and the error rate and time were estimated using 10-fold cross-validation. Not all of the algorithms were executed on the same machine and so we have employed a time normalization factor to minimize the differences.

The rankings used to test the reduction algorithm were generated with the ranking method reviewed briefly in Section 2. Three settings for the compromise between accuracy and time were tested, with $AccD \in \{0.1\%, 1\%, 10\%\}$, corresponding to growing importance of time.

Table 1. ARR scores for different values of *AccD*.

	AccD		
methods	10	100	1000
reduced	1.1225	1.0048	1.0037
top 3	1.1207	0.9999	0.9889
all	0.8606	0.9975	1.0093

The parameters used in the ranking reduction method were the following:

– Performance information for the *ten* nearest neighbors (datasets) was used to determine the redundancy of an algorithm, corresponding to approximately 20% of the total number of datasets. Although the method seems to be relatively insensitive to the number of neighbors, this percentage has obtained good results in previous experiments [8].
– An algorithm is dropped if it is not significantly better than another algorithm retained in the ranking on at least *10%* of datasets selected (i.e. one dataset, since ten neighbors are used).
– The significance of differences between two algorithms has been checked with the paired t test with a confidence level of *95%*.

We have compared our method with the full ranking and with a reduced ranking obtained by selecting the top three algorithms in the full ranking.

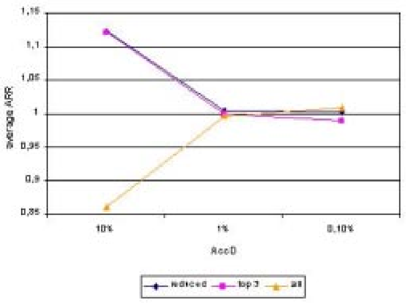


Fig. 1. Average ARR scores over 53 dataset for different values of *AccD*.

As can be seen in Table 1 and Figure 1 the method proposed always performs better than the simple reduction method that uses the top three algorithms in the full ranking.

As for the choice of executing all the algorithms, when time is given great importance, this option is severely penalized in terms of the ARR score. When accuracy is the dominant criterion, executing all the algorithms represents the option. This result is of course to be expected, since this strategy consists of

Table 2. Complete and reduced rankings for the `segment`dataset for two different values of *AccD*. The algorithms eliminated in the reduced ranking are indicated by a '*'.

	AccD			
rank	10%		1%	
1	C5.0 tree		C5.0 w/ boosting	
2	LD		C5.0 rules	
3	C5.0 rules	*	C5.0 tree	
4	C5.0 w/ boosting	*	Ltree	
5	Ltree	*	IBL	
6	IBL	*	LD	
7	NB	*	NB	*
8	RIPPER	*	RIPPER	*
9	MLP	*	MLP	
10	RBFN	*	RBFN	*

executing all algorithms with the aim of identifying the best option. So, our method is suitable if time matters at least a bit.

One interesting observation is that the method presented always obtains an average ARR larger than 1. This means that, for all compromises between accuracy and time, as defined by the *AccD* parameter, it generates recommendations that are advantageous either in terms of accuracy, or in terms of time, when compared to the average of the other two methods.

We wanted to analyze some of the reduced ranking to see how many algorithms were actually dropped and which ones. Table 2 presents the recommendation that was generated for the `segment` dataset.

When time is considered very important ($AccD = 10\%$, equivalent to a compromise between 10x speed up and a 10% drop in accuracy), only two algorithms were retained in the ranking: C5.0 tree and LD. All the others marked with * have been dropped. This result corresponds quite well to our intuitions. Both C5.0tree and linear discriminant are known to be relatively fast algorithms. It is conceivable that this does not vary from dataset to dataset[1] and hence variations in accuracy on different datasets did not affect things much.

The situation is quite different if we give more importance to accuracy. Let us consider, for instance the case when $AccD = 1\%$. Here 7 out of 10 algorithms were retained, including MLP which appeared in the 9th position in the original ranking. This is justified by the historical evidence that the method takes into account when giving this recommendation. Each of the algorithms retained must have done well at least on one dataset. It appears that mlcnb, ripper and RBFN do not satisfy this condition and hence were left out[2].

[1] Although this may not be true for datasets with an extremely large number of attributes, we have used none of those.

[2] All results were obtained with default parameter settings. However, the performance of some algorithms is highly dependent on the parameter setting used. If parameter tuning was performed, some of the algorithms that were left out could be competitive in relation to others. This issue was ignored in this study.

6 Discussion

The evaluation measure presented can be improved. We can try to determine whether the ARR measure associated with one reduced ranking is higher than (or equal to, or lower than) the ARR measure associated with another plan. In addition, it is possible to carry out statistical tests with the aim of determining whether these differences are significant. This analysis can be carried out for different setting of the constant *AccD* determining the relative importance of time. This procedure can thus be used to identify the algorithms lying on, or near the, so called, efficiency frontier [4].

Furthermore, note that this method can be used to compare reduced rankings of different lengths. However, if we wanted to evaluate not only the final outcome, but also some intermediate results, this could be done as follows. In this setting, a reduced ranking can be regarded as a *plan* to be executed sequentially. Suppose a given plan P_i consists of n_1 items (classifiers) and plan P_j of n_2 items, where, say, $n2 < n1$. The first n_2 comparisons can be carried out as described earlier, using the ARR measure. After that we simply complete the shorter plan with $n_1 - n_2$ dummy entries. Each entry just replicates the accuracy characterizing the plan. It is assumed that no time is consumed in this process.

7 Conclusions

We have presented a method to eliminate the algorithms in a ranking that are expected to be redundant. The method exploits past performance information for those algorithms. The method presented addresses one shortcoming of existing ranking methods [8,4], that don't indicate, from the ranking of candidate algorithms, which ones are really worth trying. As expected, the experiments carried out show that our method performs better than the simple strategy of selecting the top three algorithms. Also, when time is very important, our method represents a significant improvement compared to running all algorithms. It is interesting to note that when accuracy is very important, although running all algorithms is the best strategy, the difference to our method is small.

Here we have concentrated on accuracy and time. It would be interesting to take other evaluation criteria into account, e.g., interpretability of the models.

Acknowledgments We would like to thank all the METAL partners for a fruitful working atmosphere, in particular to Johann Petrak for providing the scripts to obtain the meta-data and to Jörg Keller, Iain Paterson and Helmut Berrer for useful exchange of ideas. The financial support from ESPRIT project METAL, project ECO under PRAXIS XXI, FEDER, FSE, Programa de Financiamento Plurianual de Unidades de I&D and Faculty of Economics is gratefully acknowledged.

References

1. P. Brazdil and C. Soares. A comparison of ranking methods for classification algorithm selection. In R.L. de Mántaras and E. Plaza, editors, *Machine Learning: Proceedings of the 11th European Conference on Machine Learning ECML2000*, pages 63–74. Springer, 2000.
2. W.W. Cohen. Fast effective rule induction. In A. Prieditis and S. Russell, editors, *Proceedings of the 11th International Conference on Machine Learning*, pages 115–123. Morgan Kaufmann, 1995.
3. J. Gama. Probabilistic linear tree. In D. Fisher, editor, *Proceedings of the 14th International Machine Learning Conference (ICML97)*, pages 134–142. Morgan Kaufmann, 1997.
4. J. Keller, I. Paterson, and H. Berrer. An integrated concept for multi-criteria ranking of data-mining algorithms. In J. Keller and C. Giraud-Carrier, editors, *Meta-Learning: Building Automatic Advice Strategies for Model Selection and Method Combination*, 2000.
5. R. Kohavi, G. John, R. Long, D. Mangley, and K. Pfleger. MLC++: A machine learning library in c++. Technical report, Stanford University, 1994.
6. D. Michie, D.J. Spiegelhalter, and C.C. Taylor. *Machine Learning, Neural and Statistical Classification*. Ellis Horwood, 1994.
7. R. Quinlan. *C5.0: An Informal Tutorial*. RuleQuest, 1998. http://www.rulequest.com/see5-unix.html.
8. C. Soares and P. Brazdil. Zoomed ranking: Selection of classification algorithms based on relevant performance information. In D.A. Zighed, J. Komorowski, and J. Zytkow, editors, *Proceedings of the Fourth European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD2000)*, pages 126–135. Springer, 2000.
9. C. Soares, P. Brazdil, and J. Costa. Measures to compare rankings of classification algorithms. In H.A.L. Kiers, J.-P. Rasson, P.J.F. Groenen, and M. Schader, editors, *Data Analysis, Classification and Related Methods, Proceedings of the Seventh Conference of the International Federation of Classification Societies IFCS*, pages 119–124. Springer, 2000.

Non–parametric Nearest Neighbor with Local Adaptation

Francisco J. Ferrer–Troyano, Jesús S. Aguilar–Ruiz, and José C. Riquelme

Department of Computer Science, University of Sevilla
Avenida Reina Mercedes s/n, 41012 Sevilla, Spain
{ferrer,aguilar,riquelme}@lsi.us.es

Abstract. The k–Nearest Neighbor algorithm (*k-NN*) uses a classification criterion that depends on the parameter k. Usually, the value of this parameter must be determined by the user. In this paper we present an algorithm based on the NN technique that does not take the value of k from the user. Our approach evaluates values of k that classified the *training* examples correctly and takes which classified most examples. As the user does not take part in the election of the parameter k, the algorithm is non–parametric. With this heuristic, we propose an easy variation of the *k-NN* algorithm that gives robustness with noise present in data. Summarized in the last section, the experiments show that the error rate decreases in comparison with the *k-NN* technique when the best k for each database has been previously obtained.

1 Introduction

In Supervised Learning, systems based on examples (CBR, Case Based Reasoning) are object of study and improvement from their appearance at the end of the sixties. These algorithms extract knowledge by means of inductive processes from the partial descriptions given by the initial set of examples or instances. Machine learning process is usually accomplished in two functionally different phases. In the first phase of *Training* a model of the hyperspace is created by the labelled examples. In the second phase of *Classification* the new examples are classified or labelled based on the constructed model. The classifier approached in this paper belongs to the family of the nearest neighbor algorithm (from here on *NN*) where the *training* examples are the model itself. *NN* assigns to each new query the label of its nearest neighbor among those that are remembered from the phase of *Training* (from here on the set T).

In order to improve the accuracy with noise present in data, the *k-NN* algorithm introduces a parameter k so that for each new example q to be classified the classes of the k nearest neighbors of q are considered: q will be labelled with the majority class or, in case of tie, it is randomly broken. Another alternative consists in assigning that class whose average distance is the smallest one or introducing a heuristically obtained threshold $k_1 < k$ so that the assigned class will be that with a number of associated examples greater than this threshold [12]. Extending the classification criterion, the $k\text{-}NN_{wv}$ algorithms (Nearest

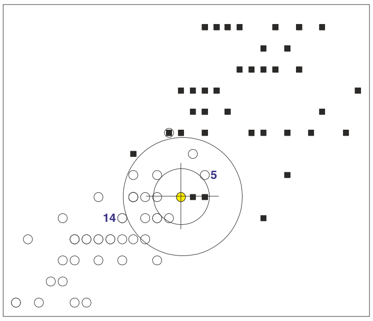


Fig. 1. The chosen value for k is decisive to classify a new example q by *k-NN* when this example is near the decision boundaries. For this example, the smaller value of the parameter k that classifies q correctly is 5.

Neighbor Weighted Voted) assign weights to the prediction made by each example. These weights can be inversely proportional to the distance with respect to the example to be classified [5,7]. Therefore, the number k of examples observed and the metric used to classify a *test* example are decisive parameters. Usually k is heuristically determined by the user or by means of cross-validation [11]. The usual metrics of these algorithms are the Euclidean distance for continuous attributes and the Overlap distance for nominal attributes (both metrics were used in our experiments).

In the last years have appeared interesting approaches that test new metrics [15] or new data representations [3] to improve accuracy and computational complexity. Nevertheless, in spite of having a wide and diverse field of application, to determine with certainty when *k-NN* obtains higher accuracy than *NN* [2] and viceversa [8] is still an open problem. In [6] it was proven that when the distance among examples with the same class is smaller than the distance among examples of different class, the probability of error for *NN* and *k-NN* tends to 0 and , respectively. But, not always this distribution for input data appears, reason why *k-NN* and *k-NN*$_{wv}$ can improve the results given by *NN* with noise present in the data.

In [13] the experimental results give rise to the two following hypotheses: a) Noisy data need large values for k; b) The performance of *k-NN* is less sensitive to the choice of a metric. Figure 1 illustrates this fact when the values of two attributes from the Iris database are projected on the plane. The X-axis measures the length of the petal and the Y-axis measures the width of the petal.

In [14] a study of the different situations in which *k-NN* improves the results of *NN* is exposed, and four classifiers are proposed (*Locally Adaptive Nearest Neighbor*, $localKNN_{ks}$) where for each new example q to be classified the parameter k takes a value k_q which is *similar* to the values that classified the M nearest neighbors e_q of q. Using a similar approach to Wettschereck's, we propose a classification criterion for new examples by taking different values for k

according to the most frequent ones that classified the original examples correctly. To calculate such values the proximity of each example to its enemy is analysed, being the *enemy* the nearest neighbor with different class. A priori, if the example to be classified is near examples having different classes among them, it might be classified by few values. But if this example is surrounded by neighbors with the same class, it could be classified by different values.

This paper presents a method that reduces and classifies according to such a criterion with no need to analyse each particular case. In this way, the impure regions and the border instances are better analysed in order to provide greater robustness with noise present in the data.

In section 2 the proposed method and their computational complexity are detailed. Section 3 describes the experiments and the results from the UCI repository [4] and section 4 summarizes the conclusions.

2 Description of the Algorithm

2.1 Approach

By means of the *k-NN* algorithm, if a new example is near the decision boundaries, the resulting class depends on the parameter k. At worst, the percentages of examples of each class are similar at these regions. In such situation, the set formed by classifying values k_{e_i} associated with each example e_i at this region can be large or zero, i.e. some examples will not have any associated value k_{e_i} which classify it correctly by *k-NN*. So, this information (the classifying values associated with the nearest neighbors of a new query q) can be not relevant to classify a new query q by *k-NN*.

We not assume that it is possible to determine the values of the parameter k which allow to classify the examples in overlapped regions. However, we assume that it is possible to improve the accuracy if several times the *k-NN* algorithm is evaluated on these regions. The idea is as simple as to give more than one opportunity to the example that is to be classified . If the example to be classified is a central example this criterion will not have any effect. If it is a border example, the accuracy can improve. Thus, the disturbing effect caused by the noise and the proximity to enemies can be smoothed. The consequences of such a bias can be explained in three cases:

- If q is a central example, the majority class might almost always be the same one for each evaluation.
- If q is a noise example, either there will not be an associated value k_q that classifies q correctly or k_q will be large.
- If q is a border example, several evaluations can avoid the errors of classification.

Figures 2 and 3 illustrate these facts by means of projections on the plane of the values of two attributes of the Horse-Colic database. In the first case (Figure 2) the value of k is slight relevant whereas in the second case (Figure 3) such

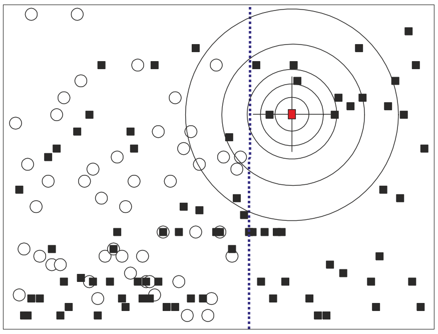


Fig. 2. Horse Colic database. If the new example to be classified is central, the majority class in each evaluation might be the same almost always.

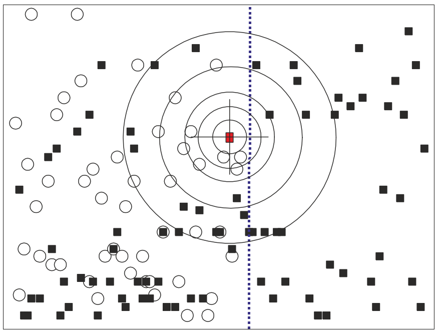


Fig. 3. Horse Colic database. If the new example to be classified is a border example, several evaluations of the *k-NN* algorithm can avoid some errors.

value is critical. Therefore, our problem is to find either the limits $[k_{q_{min}}, k_{q_{max}}]$ between which the *k-NN* algorithm will be applied for each new example q or the values, which are not necessarily continuous, $\{k_{q_1}, k_{q_2}, \ldots\}$ from which is calculated the majority class. The method has been denominated fNN (k–Frequent Nearest Neighbors) since it takes the most frequent values of k among those that classified correctly the examples of each database. In this process, there are no parameters given by the user since these values are calculated locally for each database.

2.2 The Algorithm

Let n be the number of examples of the *Training* set T. Let $kNN(e, i)$ the i^{th} nearest neighbor of an example e within T. We denote $majorityClass(e, i..j)$ as the majority class between the i^{th} and the j^{th} neighbors of e. *fNN* associates with each example e_i two values:

1. $kCMin_i$: The smallest k that classifies correctly the example e_i by using the *k-NN* algorithm, such that (see Figure 4):

$$\forall j \in [1, kCMin_i) \mid Class\,(kNN\,(e_i, j)) = Class\,(e_i) \Rightarrow \\ \Rightarrow majorityClass\,(e_i, 1..j) \neq Class\,(e_i) \qquad (1)$$

2. $kCMax_i$: If $kCMin_i$ was found, then $kCMax_i \geq kCMin_i$ and:

$$\forall j \in [kCMin_i, kCMax_i] \Rightarrow Class\,(e_i) = Class\,(kNN\,(e_i, j)) \qquad (2)$$

With these values, a new value $kLim$ is calculated which satisfies the following property:

$$\forall e_i \in T, j \in [\min(kCMin_i), kLim] \Rightarrow \exists e_k \in T \mid kCMin_k = j \qquad (3)$$

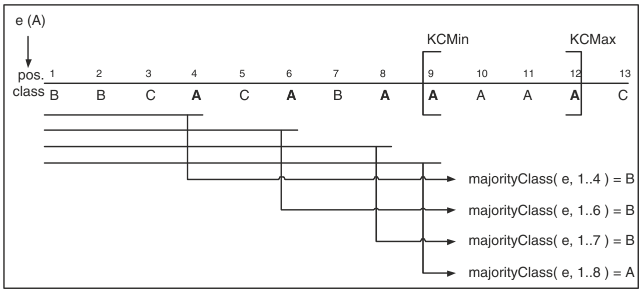

e (A)
KCMin
KCMax
pos. 1 2 3 4 5 6 7 8 9 10 11 12 13
class B B C A C A B A A A A A C
majorityClass(e, 1..4) = B
majorityClass(e, 1..6) = B
majorityClass(e, 1..7) = B
majorityClass(e, 1..8) = A

Fig. 4. Example of $kCMin$ and $kCMax$ for an example e with class A. The ties are broken with the nearest class. Majority class is not A from $k = 1$ to $k = 8$. When $k = 9$ the majority class is A ($kCMin_e = 9$). All the examples have class A from $k = 9$ to $k = 12$. Finally, the thirteenth neighbor of e has class C, so that $kCMax_e = 12$.

where $min(kCMin_i)$ is the least $kCMin_i$ value of each example.

Those examples e_i that either have not any associated value $kCMin_i$ or have an associated value $kCMin_i > kLim$ are considered outliers and they are removed. The resulting reduced set (from here on T_f) will be used as *Training* model for our algorithm.

In order to classify a new example q, the *k-NN* algorithm is applied several times to the same example q by varying the value of k. These values belong the interval $[\min(kCMin_i), kLim]$. The assigned label will be that among the nearest to q that is most frequent in the k_T evaluations of *k-NN*. Thus, the computational complexity of *fNN* is: $\Theta\left(n^2 \cdot (\log n + 1) + n \cdot (\log n + \delta + 2) + kLim^2\right)$.

In the first phase the set T_f is generated. The $n - 1$ nearest neighbors for the n examples of the *Training* set are ordered ($\Theta\left(n \cdot (n + n \cdot \log n)\right)$). So, for each example it is computed the distance to all the neighbors ($\Theta(n)$) and then the associated list is ordered ($\Theta(n \log n)$). After this procedure $kLim$ is found ($\Theta(n \cdot \delta)$, $\min(kCMin_i) \leq \delta \leq kLim$) and the outliers are removed ($\Theta(n)$).

In the second phase a *test* example is classified ($\Theta\left(n \cdot (\log n + 1) + kLim^2\right)$). The pseudo code for the *fNN* algorithm is shown in Figure 5 where n is the original number of examples and c the number of different labels for the class.

3 Results

To carry out the method and the test, the Euclidean distance for continuous attributes and the Overlap distance for the nominal attributes were used. The values of the continuous attributes were normalized in the interval [0,1]. Examples with missing-class was removed and attributes with missing-values was treated with the mean or mode, respectively. *fNN* was tested on 20 databases from the Machine Learning Database Repository at the University of California, Irvine [4]. In order to reduce statistical variation, each experiment was executed

```
function fNN(Train:SET; query:INSTANCE) return (label:Integer)
var
  T_f: Set;  k,k_Lim,label: Integer;  k_Vect: VECTOR[n][2] Of Integer
  frequencies: VECTOR[c] Of Integer
beginF
  // Calculating kCMin, kCMax for each example.
  Calculate_MinMax_Values_K(Train, k_Vect)
  // Finding the limits for the evaluations of kNN.
  Calculate_Limits(k_Vect,k_Lim)
  // Removing the examples whose kCMin does not belong to [1,k_Lim].
  T_f:= Delete_Outliers(Train, k_Lim)
  for k:= Min(kCMin) to k_Lim
   label:= Classify_KNN(query,T_f,k)
   frequencies[label]:= frequencies[label]+ 1/Average_Dist(query,label,k)
  return(frequencies.IndexOfMaxElement())
endF
```

Fig. 5. Pseudo code for *fNN*.

by means of 10-folds cross-validation. *fNN* was compared with *k-NN* using 25 different values of k (the odd numbers belonging to interval [1, 51]). This limit was fixed after observing for all databases how the accuracy decreased from a value near the best k for each database (being 33 the maximum value for *Heart Cleveland*) database. In Table 1 is reported the main results obtained. The average-accuracy with the associated standard deviation and the computational cost by means of *fNN* is showed in Columns 2a and 2b respectively. The *k-NN* algorithm is included for comparison using the best k for each database (Column 3a) and the best average-value (k=1) for all databases (Column 1a). Both computational cost for *k-NN* were very similar and they are showed in Column 1b. Column 3b shows the best value of k for each database by *k-NN*. Column 2c show the size of T_f regarding the *Training* set and Column 2d show the values of $kLim$ for each database, i.e. the limit for k by *fNN*. The databases marked with * mean an improvement of *fNN* regarding *1-NN* by means of t-Student statical test using $\alpha = 0.05$. We can observe in Table 1 that *fNN* obtained better precision than *1-NN* for 13 databases where the best k for the *k-NN* algorithm was a high value, so that:

- If $KLim < k_{best}$ for *k-NN*, then *fNN* provides higher accuracy than *1-NN*.
- The percentage of examples that are excluded from T_f is a minimum error bound for *k-NN*.

4 Conclusions

An easy variation of the *k-NN* algorithm has been explained and evaluated in this paper. Experiments with commonly used databases indicate that exits do-

Table 1. Results for 20 databases from the UCI repository by using 10-folds cross-validation. Column 1 shows the average accuracy with the standard deviation and the computational cost by *k-NN* with k=1 (the best value for all databases). Column 2 shows the same percentages obtained by *fNN*, the percentage of examples retained from the *Training* set and the value of kLim, i.e. the limit for *k* by *fNN* . Column 3 shows the best accuracy with the standard deviation by the *k-NN* algorithm when the best *k* is found. This best *k* was looked for in the odd numbers belonging to interval [1,51].

	1-NN		**fNN**				**best k-NN**	
Domain	Pred. Acc.	Time	Pred. Acc.	Time	%Ret.	kLim	Pred. Acc.	best k
Anneal	91.76 ±2.4	1.10	90.32 ±1.8	15.7	96.5	9	91.76 ±2.4	1
Balance Scale*	77.76 ±4.8	0.25	89.44 ±1.5	4.0	89.7	11	89.76 ±1.4	21
B. Cancer (W)	95.56 ±2.2	0.20	96.57 ±1.7	2.91	92.1	9	96.85 ±2.0	17
Credit Rating*	80.72 ±2.2	0.56	87.11 ±1.8	7.59	91.9	7	87.68 ±1.6	13
German Credit	72.29 ±2.9	1.42	74.61 ±3.4	18.2	89.1	21	73.09 ±4.2	17
Glass	70.19 ±2.0	0.04	69.16 ±1.3	0.64	84.4	9	70.19 ±2.0	1
Heart D. (C)*	74.92 ±2.5	0.09	81.52 ±2.0	1.32	89.3	13	83.17 ±2.7	33
Hepatitis*	81.29 ±0.8	0.03	87.11 ±0.9	0.43	88.5	9	85.16 ±1.0	7
Horse Colic	67.93 ±3.0	0.22	69.02 ±1.6	2.90	83.7	11	70.38 ±2.6	7
Ionosphere	86.61 ±1.9	0.29	85.18 ±2.0	3.66	91.1	13	86.61 ±1.9	1
Iris	95.33 ±0.8	0.01	96.00 ±0.8	0.29	95.6	1	97.33 ±0.8	15
Pima Diabetes	71.21 ±5.0	0.56	74.09 ±3.9	7.62	87.7	15	75.52 ±4.2	17
Primary Tumor*	35.69 ±1.3	0.05	42.19 ±1.5	0.81	54.9	11	43.07 ±1.5	29
Sonar	86.54 ±1.5	0.16	86.06 ±1.2	1.91	94.1	5	86.54 ±1.5	1
Soybean	90.92 ±3.5	0.65	91.07 ±2.9	8.42	95.9	13	90.92 ±3.5	1
Vehicle	70.06 ±3.0	1.12	69.97 ±2.9	13.5	89.9	13	70.06 ±3.0	1
Voting	92.18 ±1.6	0.07	92.64 ±1.3	1.18	95.1	3	93.56 ±1.0	5
Vowel	99.39 ±0.9	1.19	98.79 ±1.3	16.1	99.1	1	99.39 ±0.9	1
Wine	96.07 ±1.1	0.03	96.63 ±0.7	0.54	98.1	5	97.75 ±0.7	31
Zoo*	97.03 ±0.6	0.01	93.07 ±1.1	0.19	95.6	1	97.03 ±0.6	1
Average	81.67 ±2.2	0.40	83.53 ±1.8	5.39	90.11	9	84.29 ±2.0	11

mains where classification is very sensitive to the parameter *k* by using the *k-NN* algorithm. For these input data, we could summarize several aspects:

– Without the need of parameter, *fNN* is a reduction and classification technique that keeps the average accuracy of the *k-NN* algorithm.
– *kLim* and the size of T_f compared to the size of T are an approximated indicator for the percentage of examples that cannot be correctly classified by the *k-NN* algorithm.
– The reduction of the database is very similar to the reduction that makes *CNN* [15], so that *fNN* is less restrictive than *CNN*. With large databases, this reduction can accelerate the learning process for the *k-NN* algorithm.

5 Future Work

Actually we are testing *fNN* with other classifiers. Particularly, we have chosen two systems, C4.5 [9] and HIDER [10], which generate decision trees and axis–parallel decision rules, respectively. Due to *fNN* makes a previous reduction, we have chosen the method EOP [1], which reduces databases conserving the decision boundaries that are parallel to the axis.

Acknowledgments

The research was supported by the Spanish Research Agency CICYT under grant TIC99-0351. We thank the anonymous reviewers for providing helpful advice and suggestions that improved this work.

References

1. J. S. Aguilar, J. C. Riquelme, and M. Toro. Data set editing by ordered projection. In *Proceedings of the* 14^{th} *European Conference on Artificial Intelligence*, pages 251–255, Berlin, Germany, August 2000.
2. D. W. Aha, D. Kibler, and M. K. Albert. Instance-based learning algorithms. *Machine Learning*, 6:37–66, 1991.
3. S. Arya, D. M. Mount, N. S. Netanyahu, R. Silverman, and A. Wu. An optimal algorithm for nearest neighbor searching. In *Proceedings of 5th ACM SIAM Symposium on discrete Algorithms*, pages 573–582, 1994.
4. C. Blake and E. K. Merz. Uci repository of machine learning databases, 1998.
5. S. Cost and S. Salzberg. A weighted nearest neighbor algorithm for learning with symbolic features. *Machine Learning*, 10:57–78, 1993.
6. T. M. Cover and P. E. Hart. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, IT-13(1):21–27, 1967.
7. S.A. Dudani. The distance-weighted k-nearest-neighbor rule. *IEEE Transactions on Systems, Man and Cybernetics*, SMC-6, 4:325–327, 1975.
8. R. C. Holte. Very simple classification rules perform well on most commonly used datasets. *Machine learning*, 11:63–91, 1993.
9. J. R. Quinlan. *C4.5: Programs for machine learning.* Morgan Kaufmann, San Mateo, California, 1993.
10. J. C. Riquelme, J. S. Aguilar, and M. Toro. Discovering hierarchical decision rules with evolutive algorithms in supervised learning. *International Journal of Computers, Systems and Signals*, 1(1):73–84, 2000.
11. M. Stone. Cross-validatory choice and assessment of statistical predictions. *Journal of the Royal Statistical Society B*, 36:111–147, 1974.
12. I. Tomek. An experiment with the edited nearest-neighbor rule. *IEEE Transactions on Systems, Man and Cybernetics*, 6(6):448–452, June 1976.
13. C. Wettschereck. *A Study of Distance-Based Machine Learning Algorithms.* PhD thesis, Oregon State University, 1995.
14. D. Wettschereck and T.G. Dietterich. Locally adaptive nearest neighbor algorithms. *Advances in Neural Information Processing Systems*, (6):184–191, 1994.
15. D. R. Wilson and T. R. Martinez. Improved heterogeneous distance functions. *Journal of Artificiall Intelligence Research*, 6(1):1–34, 1997.

Selection Restrictions Acquisition from Corpora

Pablo Gamallo[⋆], Alexandre Agustini [⋆⋆], and Gabriel P. Lopes

CENTRIA, Departamento de Informática Universidade Nova de Lisboa, Portgual
{gamallo,aagustini,gpl}@di.fct.unl.pt

Abstract. This paper describes an automatic clustering strategy for acquiring selection restrictions. We use a knowledge-poor method merely based on word cooccurrence within basic syntactic constructions; hence, neither semantic tagged corpora nor man-made lexical resources are needed for generalising semantic restrictions. Our strategy relies on two basic linguistic assumptions. First, we assume that two syntactically related words impose semantic selectional restrictions to each other (*co-specification*). Second, it is also claimed that two syntactic contexts impose the same selection restrictions if they cooccur with the same words (*contextual hypothesis*). In order to test our learning method, preliminary experiments have been performed on a Portuguese corpus.

1 Introduction

The general aim of this paper is to describe a particular corpus-based method for semantic information extraction. More precisely, we implement a knowledge-poor system that uses syntactic information to acquire selection restrictions and semantic preferences constraining word combination.

According to Gregory Grefenstette [9,10], knowledge-poor approaches use no presupposed semantic knowledge for automatically extracting semantic information. They are characterised as follows: no domain-specific information is available, no semantic tagging is used, and no static sources as machine readable dictionaries or handcrafted thesauri are required. Hence, they differ from knowledge-rich approaches in the amount of linguistic knowledge they need to activate the semantic acquisition process. Whereas knowledge-rich approaches require previously encoded semantic information (semantic tagged corpora and/or man-made lexical resources [17,6,1]), knowledge-poor methods only need a coarse-grained notion of linguistic information: word cooccurrence. In particular, the main aim of knowledge-poor approaches is to calculate the frequency of word cooccurrences within either syntactic constructions or sequences of n-grams in order to extract semantic information such as selection restrictions [19,11,3], and word ontologies [12,15,9,13]. Since these methods do not require previously defined semantic knowledge, they overcome the well-known drawbacks associated with handcrafted thesauri and supervised strategies.

[⋆] Research supported by the PRAXIS XXI project, FCT/MCT, Portugal
[⋆⋆] Research sponsored by CAPES and PUCRS - Brazil

Nevertheless, our method differs from standard knowledge-poor strategies on two specific issues: both the way of extracting word similarity and the way of defining syntactic contexts. We use the contextual hypothesis to characterise word similarity and the co-specification hypothesis to define syntactic contexts.

1.1 The Contextual Hypothesis

In most knowledge-poor approaches to selection restriction learning, the process of inducing and generalising semantic information from word cooccurrence frequencies consists in automatically clustering words considered as similar. The best-known strategy for measuring word similarity is based on Harris' *distributional hypothesis*. According to this assumption, words cooccurring in similar syntactic contexts are semantically similar and, then, should be clustered into the same semantic class. However, the learning methods based on the distributional hypothesis may lead to cluster in the same class words that fill different selection restrictions. Let's analyse the following examples taken from [20]:

(a) John worked till late at the *council*
(b) John worked till late at the *office*
(c) the *council* stated that they would raise taxes
(d) the *mayor* stated that he would raise taxes

On the basis of the distributional hypothesis, since *council* behaves similarly to *office* and *mayor* they would be clustered together into the same word class. Nevertheless, the bases for the similarity between *council* and *office* are different from those relating *council* and *mayor*. Whereas *council* shares with *office* syntactic contexts associated mainly with LOCATIONS (e.g., the argument of *work at* in phrases (a) and (b)), *council* shares with *mayor* contexts associated with AGENTS (e.g., the subject of *state* in phrases (c) and (d)). That means that a polysemous word like *council* should be clustered into various semantic word classes, according to its heterogeneous syntactic distribution. Each particular sense of the word is related to a specific type of distribution. Given that the clustering methods based on the distributional hypothesis solely take into account the global distribution of a word, they are not able to separate and acquire its different contextual senses.

In order to extract contextual word classes from the appropriate syntactic constructions, we claim that similar syntactic contexts share the same semantic restrictions on words. Instead of computing word similarity on the basis of the too coarse-grained distributional hypothesis, we measure the similarity between syntactic contexts in order to identify common selection restrictions. More precisely, we assume that two syntactic contexts occurring with (almost) the same words are similar and, then, impose the same semantic restrictions on those words. That is what we call *contextual hypothesis*. Semantic extraction strategies based on the contextual hypothesis may account for the semantic variance of words in different syntactic contexts. Since these strategies are concerned with the extraction of semantic similarities between syntactic contexts, words will be clustered with regard to their specific syntactic distribution. Such clusters represent context-dependent semantic classes. Except the cooperative system Asium introduced in [4,5,2], few or no research on semantic extraction have been based on such a hypothesis.

1.2 Co-specification

Traditionally, a binary syntactic relationship is constituted by both the word that imposes linguistic constraints (the predicate) and the word that must fill such constraints (its argument). In a syntactic relationship, each word plays a fixed role. The argument is perceived as the word specifying or modifying the syntactic-semantic constraints imposed by predicate, while the latter is viewed as the word specified or modified by the former. However, recent linguistic research assumes that the two w ords related by a syntactic dependency are mutually specified [16,8]. Each word imposes semantic conditions on the other word of the dependency, and each word elaborates them. Consider the relationship between the polysemic verb *load* and the polysemic noun books in the non ambiguous expression *to load the books.* On the one hand, the polysemic verb *load* conveys at least two alternate meanings: "bringing something to a location" (e.g., *Ann loaded the hay onto the truck*), and "modifying a location with something" (e.g., *Ann loaded the truck with the hay*). This verb is disambiguated by taking into account the sense of the words with which it combines within the sentence. On the other hand, the noun *book(s)* is also a polysemic expression. Indeed, it refers to different types of entities: "physical objects" (*rectangular book*), and "symbolic entities" (*interesting book*). Yet, the constraints imposed by the words with which it combines allows the noun to be disambiguated. Whereas the adjective *rectangular* activates the physical sense of *book*, the adjective *interesting* makes reference to its symbolic content.

In *to load the books*, the verb *load* activates the physical sense of the noun, while *books* leads *load* to refer to the event of bringing something to a location. The interpretation of the complex expression is no more ambiguous. Both expressions, *load* and *books*, cooperate to mutually restrict their meaning. The process of mutual restriction between two related words is called by Pustejovsky "co-specification" or "co-composition" [16]. Co-specification is based on the following idea. Two syntactically dependent expressions are no longer interpreted as a standard pair "predicate-argument", where the predicate is the active function imposing the semantic preferences on a passive argument, which matches such preferences. On the contrary, each word of a binary dependency is perceived simultaneously as a predicate and an argument. That is, each word both imposes semantic restrictions and matches semantic requirements. When one word is interpreted as an active functor, the other is perceived as a passive argument, and conversely. Both dependent expressions are simultaneously active and passive compositional terms. Unlike most work on selection restrictions learning, our notion of "predicate-argument" frame relies on the active process of semantic co-specification, and not on the trivial operation of argument specification. This trivial operation only permits the one-way specification and disambiguation of the argument by taking into account the sense of the predicate. Specification and disambiguation of the predicate by the argument is not considered.

In this paper, we describe a knowledge-poor unsupervised method for acquiring selection restrictions, which is based on the contextual and the co-specification hypotheses. The different parts of this method will be outlined in the next section.

2 System Overview

To evaluate the hypotheses presented above, a software package was developed to support the automatic acquisition of semantic restrictions. The system is constituted by four related modules, illustrated in Figure 1. In the following paragraphs we merely outline the overall functionalities of these modules. Then, in the remainder of the paper, we describe accurately the specific objects and processes of each module.

Parsing: The raw text is tagged [14] and partially analysed [18]. Then, an attachment heuristic is used to identify *binary dependencies*. The result is a list of cooccurrence triplets containing the syntactic relationship and the lemmas of the two related head words. This module will be described in section 3.1

Extracting: The binary dependencies are used to extract the *syntactic contexts*. Unlike most work on selection restrictions learning, the characterisation of syntactic contexts relies on the dynamic process of co-specification. Then, the word sets that appear in those contexts are also extracted. The result is a list of *contextual word sets*. This module will be analysed in section 3.2.

Filtering: Each pair of contextual word sets are statistically compared using a variation of the weighted Jaccard Measure [9]. For each pair of contextual sets considered as similar, we select only the words that they share. The result is a list of semantically homogenous word sets, called *basic classes*. Section 4.1 describes this module.

Clustering: Basic classes are successively aggregated by a conceptual clustering method to induce more general classes, which represent extensionally the selection restrictions of syntactic contexts. We present this module in section 4.2. Finally, the classes obtained by clustering are used to update the subcategorisation information in the dictionary.[1]

The system was tested over the Portuguese text corpora *P.G.R.*[2]. Some results are analysed in section 4.3. The fact of using specialised text corpora makes

[1] The first tasks, tagging and parsing, are based on a non domain-specific dictionary for Portuguese, which merely contains morphosyntactic information. The subcategorisation restrictions extracted by our method are used to extend the morphosyntactic information of the dictionary.

[2] P.G.R. (*Portuguese General Attorney Opinions*) is constituted by case-law documents.

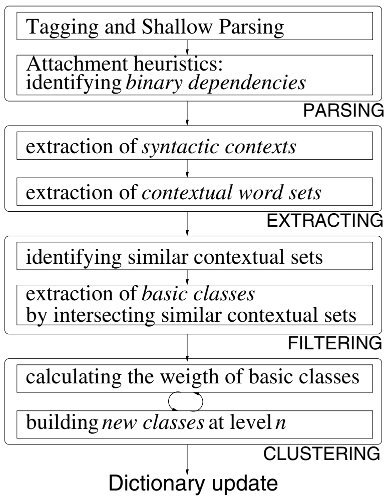

Tagging and Shallow Parsing
Attachment heuristics: identifying *binary dependencies*
PARSING
extraction of *syntactic contexts*
extraction of *contextual word sets*
EXTRACTING
identifying similar contextual sets
extraction of *basic classes* by intersecting similar contextual sets
FILTERING
calculating the weigth of basic classes
building *new classes* at level *n*
CLUSTERING
Dictionary update

Fig. 1. System modules

easier the learning task, given that we have to deal with a limited vocabulary with reduced polysemy. Furthermore, since the system is not dependent of a specific language such as Portuguese, it could be applied to whatever natural language.

3 Identification of Binary Dependencies and Extraction of Syntactic Contexts

Binary dependencies and syntactic contexts are directly associated with the notion of selection restrictions. Selection restrictions are the semantic constraints that a word needs to match in order to be syntactically dependent and attached to another word. According to the co-specification hypothesis, two dependent words can be analysed as two syntactic contexts of specification. So, before describing how selection restrictions are learned, we start by defining first how binary dependencies are identified, and second how syntactic contexts are extracted from binary dependencies.

3.1 Binary Dependencies

We assume that basic syntactic contexts are extracted from binary syntactic dependencies. We use both a shallow syntactic parser and a particular attachment heuristic to identify binary dependencies. The parser produces a single partial syntactic description of sentences, which are analysed as sequences of basic chunks (NP, PP, VP, . . .). Then, attachment is temporarily resolved by a simple heuristic based on right association (a chunk tend to attach to another chunk immediately to its right). Finally, we consider that the word heads of two attached chunks form a binary dependency. It can be easily seen that syntactic errors may appear since the attachment heuristic does not take into account distant dependencies.[3] For reasons of attachment errors, it is argued here that the binary dependencies identified by our basic heuristic are mere hypotheses on attachment; hence they are mere candidate dependencies. Candidate dependencies will be checked by taking into account further information on the attached words. In particular, a candidate dependency will be checked and finally verified only if the two attached words impose selection restrictions to each other. Therefore, the test confirming or not a particular attachment relies on the semantic information associated with the related words. Let's describe first the internal structure of a candidate dependency between two words.

A candidate syntactic dependency consists of two words and the hypothetical grammatical relationship between them. We represent a dependency as the following binary predication: $(r; w1^{\downarrow}, w2^{\uparrow})$. This binary predication is constituted by the following entities:

– the binary predicate r, wich can be associated to specific prepositions, subject relations, direct object relations, etc.;
– the roles of the predicate, "$^{\downarrow}$" and "$^{\uparrow}$", which represent the *head* and *complement* roles, respectively;
– the two words holding the binary relation: $w1$ and $w2$.

Binary dependencies denote grammatical relationships between the head and its complement. The word indexed by "$^{\downarrow}$" plays the role of *head*, whereas the word indexed by "$^{\uparrow}$" plays the role of *complement*. Therefore, $w1$ is perceived as the head and $w2$ as the complement.

Furthermore, the binary dependencies (i.e., grammatical relationships) we have considered are the following: subject (noted *subj*), direct object (noted *dobj*), prepositional object of verbs, and prepositional object of nouns, both noted by the specific preposition.

3.2 Extraction of Syntactic Contexts and Co-specification

Syntactic contexts are abstract configurations of specific binary dependencies. We use λ-abstraction to represent the extraction of syntactic contexts. A syntactic context is extracted by λ-abstracting one of the related words of a binary dependency. Thus, two complementary syntactic context can be λ-abstracted from the binary predication associated with a syntactic dependency: $[\lambda x^{\downarrow}(r; x^{\downarrow}, w2^{\uparrow})]$ and $[\lambda x^{\uparrow}(r; w1^{\downarrow}, x^{\uparrow})]$.

The syntactic context of word w2, $[\lambda x^{\downarrow}(r; x^{\downarrow}, w2^{\uparrow})]$, can be defined extensionally as the set of words that are the *head* of $w2$. The exhaustive enumeration of every word that can occur with that syntactic frame enables us to characterise extensionally its selection restrictions. Similarly, The syntactic context of word

[3] The errors are caused, not only by the too restrictive attachment heuristic, but also by further misleadings, e.g., words missing from the dictionary, words incorrectly tagged, other sorts of parser limitations, etc.

$w1$, $[\lambda x^{\uparrow}(r; w1^{\downarrow}, x^{\uparrow})]$, represents the set of words that are *complement* of $w1$. This set is perceived as the extensional definition of the selection restrictions imposed by the syntactic context. Consider Table 1. The left column contains expressions constituted by two words syntactically related by a particular type of syntactic dependency. The right column contains the syntactic contexts extracted from these expressions. For instance, from the expression `presidente da república` (*president of the republic*), we extract two syntactic contexts: both $[\lambda x^{\downarrow}(de; x^{\downarrow}, república^{\uparrow})]$, where `república` plays the role of *complement*, and $[\lambda x^{\uparrow}(de; presidente^{\downarrow}, x^{\uparrow})]$, where `presidente` is the *head.*

Table 1. Syntactic contexts extracted from binary expressions

Binary Expressions	Syntactic Contexts
`presidente da república` (*president of the republic*)	$[\lambda x^{\downarrow}(de; x^{\downarrow}, república^{\uparrow})]$, $[\lambda x^{\uparrow}(de; presidente^{\downarrow}, x^{\uparrow})]$
`nomeação do presidente` (*nomination for president*)	$[\lambda x^{\downarrow}(de; x^{\downarrow}, presidente^{\uparrow})]$, $[\lambda x^{\uparrow}(de; nomeação^{\downarrow}, x^{\uparrow})]$
`nomeou o presidente` (*nominated the president*)	$[\lambda x^{\downarrow}(dobj; x^{\downarrow}, presidente^{\uparrow})]$, $[\lambda x^{\uparrow}(dobj; nomear^{\downarrow}, x^{\uparrow})]$
`discutiu sobre a nomeação` (*disscussed about the nomination*)	$[\lambda x^{\downarrow}(sobre; x^{\downarrow}, nomeação^{\uparrow})]$, $[\lambda x^{\uparrow}(sobre; discutir^{\downarrow}, x^{\uparrow})]$

Since syntactic configurations impose specific selectional preferences on words, the words that match the semantic preferences (or selection restrictions) required by a syntactic context should constitute a semantically homogeneous word class. Consider the two contexts extracted from `presidente da república`. On the one hand, context $[\lambda x^{\uparrow}(de; presidente^{\downarrow}, x^{\uparrow})]$ requires a particular noun class, namely human organizations. In corpus P.G.R., this syntactic context selects for nouns such as `república` (*republic*), `governo` (*government*), `instituto` (*institute*), conselho (*council*),... On the other hand, context $[\lambda x^{\downarrow}(r; x^{\downarrow}, república^{\uparrow})]$ requires nouns denoting either human beings or organizations: `presidente` (*president*), `ministro` (*minister of state*), `assembleia` (*assembly*), `governo`, (*government*) `procurador` (*attorney*), `procuradoria-geral` (*attorneyship*) , `ministério` (*state department*), etc.

It follows that the two words related by a syntactic dependency are mutually determined. The context constituted by a word and a specific function imposes semantic conditions on the other word of the dependency. The converse is also true. As has been said, the process of mutual restriction between two related words is called co-specification. In `presidente da república`, the context constituted by the noun `presidente` and the grammatical function *head* somehow restricts the sense of `república`. Conversely, both the noun `república` and the role of *complement* also restrict the sense of `presidente`:

– $[\lambda x^{\downarrow}(de; x^{\downarrow}, república^{\uparrow})]$ selects for `presidente`
– $[\lambda x^{\uparrow}(de; presidente^{\downarrow}, x^{\uparrow})]$ selects for `república`

In our system, the extraction module consists of the two following tasks: first, the syntactic contexts associated with all candidate binary dependencies are extracted. Then, the set of words appearing in those syntactic contexts are selected. The words appearing in a particular syntactic context form a *contextual word set*. Contextual word sets are taken as the input of the processes of filtering and clustering; these processes will be described in the next section.

Let's note finally that unlike the Grefenstette's approach [9], information on co-specification is available for the characterisation of syntactic contexts. In [7], a strategy for measuring word similarity based on the co-specification hypothesis was compared to the Grefensetette's strategy. Experimental tests demonstrated that co-specification allows a finer-grained characterisation of syntactic contexts.

4 Filtering and Clustering

According to the contextual hypothesis introduced above, two syntactic contexts that select for the same words should have the same extensional definition and, then, the same selection restrictions. So, if two contextual word sets are considered as similar, we infer that their associated syntactic contexts are semantically similar and share the same selection restrictions. In addition, we also infer that these contextual word sets are semantically homogeneous and represent a contextually determined class of words. Let's take the two following syntactic contexts and their associated contextual word sets:

$$\begin{aligned}&\left[\lambda x^{\uparrow}(of; infringement^{\downarrow}, x^{\uparrow})\right] = \{article\ law\ norm\ precept\ statute\ \ldots\}\\&\left[\lambda x^{\uparrow}(dobj; infringe^{\downarrow}, x^{\uparrow})\right] = \{article\ law\ norm\ principle\ right\ \ldots\}\end{aligned}$$

Since both contexts share a significant number of words, it can be argued that they share the same selection restrictions. Furthermore, it can be inferred that their associated contextual sets represent the same context-dependent semantic class. In our corpus, context $[\lambda x^{\uparrow}(dobj; violar^{\downarrow}, x^{\uparrow})]$ (*to infringe*) is not only considered as similar to context $[\lambda x^{\downarrow}(dobj; violação^{\downarrow}, x^{\uparrow})]$ (*infringemen t of*), but also to other contexts such as: $[\lambda x^{\downarrow}(dobj; respeitar^{\downarrow}, x^{\uparrow})]$ (*to respect*) and $[\lambda x^{\uparrow}(dobj; aplicar^{\downarrow}, x^{\uparrow})]$ (*to apply*).

In this section, we will specify the procedure for learning context-dependent semantic classes from the previously extracted contextual sets. This will be done in two steps:

– Filtering: word sets are automatically cleaned by removing those words that are not semantically homogenous.
– Conceptual clustering: the previously cleaned sets are successively aggregated into more general clusters. This allows us to build more abstract semantic classes and, then, to induce more general selection restrictions.

4.1 Filtering

As has been said in the introduction, the cooperative system Asium is also based on the contextual hypothesis [4,5]. This system requires the interactive participation of a language specialist in order to filter and clean the word sets when they are taken as input of the clustering strategy. Such a cooperative method proposes to manually remove from the sets those words that have been incorrectly tagged or analysed. Our strategy, by contrast, intends to automatically remove incorrect words from sets. Automatic filtering consists of the following subtasks:

First, each word set is associated with a list of its most similar sets. Intuitively, two sets are considered as similar if they share a significant number of words. Various similarity measure coefficients were tested to create lists of similar sets. The best results were achieved using a particular weighted version of the Jaccard coefficient, where words are weighted considering their dispersion (global weight) and their relative frequency for each context (local weight). Word dispersion (global weight) *disp* takes into account how many different contexts are associated with a given word and the word frequency in the corpus. The local weight is calculated by the relative frequency fr of the pair word/context. The weight of a word with a context *cntx* is computed by the following formula:

$$W(word_i, cntx_j) = log_2(fr_{ij}) * log_2(disp_i)$$

where

$$fr_{ij} = \frac{frequency\, of\, word_i\, with\, cntx_j}{sum\, of\, frequencies\, of\, words\, occurring\, in\, cntx_j}$$

and

$$disp_i = \frac{\sum_j frequency\, of\, word_i\, with\, cntx_j}{number\, of\, contexts\, with\, word_i}$$

So, the weighted Jaccard similarity WJ between two contexts m and n is computed by[4]:

$$WJ(cntx_m, cntx_n) = \frac{\sum_{common_i}(W(cntx_m, word_i) + W(cntx_n, word_i))}{\sum_j(W(cntx_m, word_j) + W(cntx_n, word_j))}$$

Then, once each contextual set has been compared to the other sets, we select the words shared by each pair of similar sets, i.e., we select the intersection between each pair of sets considered as similar. Since words that are not shared by two similar sets could be incorrect words, we remove them. Intersection allows us to clear sets of words that are not semantically homogenous. Thus, the intersection of two similar sets represents a semantically homogeneous class, which we call *basic class*. Let's take an example. In our corpus, the most similar set to $[\lambda x^{\uparrow}(de; violação^{\downarrow}, x^{\uparrow})]$ (*infringement of*)) is $[\lambda x^{\uparrow}(dobj; violar^{\downarrow}, x^{\uparrow})]$ (*infringe*) . Both sets share the following words:

```
sigilo princípios preceito plano norma lei estatuto disposto
disposição direito convenção artigo
```
(secret principle precept plan norm law statute rule precept
right convention article)

[4] *common* means that just common words to both contexts m and n are computed

This basic class does not contain incorret words such as `vez, flagrante-mente, obrigação, interesse` (*time, notoriously, obligation, interest*), which were oddly associated to the context $[\lambda x^{\uparrow}(de; \textit{violação}^{\downarrow}, x^{\uparrow})]$, but which does not appear in context $[\lambda x^{\uparrow}(dobj; violar^{\downarrow}, x^{\uparrow})]$. This class seems to be semantically homogenous because it contains only words referring to legal documents. Once basic classes have been created, they are used by the conceptual clustering algorithm to build more general classes. Note that this strategy do not remove neither infrequent nor very frequent words. Frequent and infrequent words may be semantic significant provided that they occur with similar syntactic contexts.

4.2 Conceptual Clustering

We use an agglomerative (bottom-up) clustering for successivelly aggregating the previously created basic classes. Unlike most research on conceptual clustering, aggregation does not rely on a statistical distance between classes, but on empirically set conditions and constraints [21]. These conditions will be discussed below. Figure 2 shows two basic classes associated with two pairs of similar syntactic contexts. $[CONTX_i]$ represents a pair of syntactic contexts sharing the words `preceito, lei, norma` (*precept, law, norm*, and $[CONTX_j]$ represents a pair of syntactic contexts sharing the words `preceito, lei, direito` (*precept, law, right*). Both basic classes are obtained from the filtering process described in the previous section. Figure 3 illustrates how basic classes are aggregated into more general clusters. If two classes fill the conditions that we will define later, they can be merged into a new class. The two basic classes of the example are clustered into the more general class constituted by `preceito, lei, norma, direito`. Such a generalisation leads us to induce syntactic data that does not appear in the corpus. Indeed, we induce both that the word `norma` may appear in the syntactic contexts represented by $[CONTX_j]$, and that the word `direito` may be attached to the syntactic contexts represented by $[CONTX_i]$.

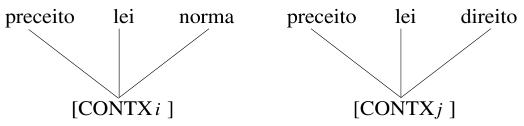


Fig. 2. Basic classes

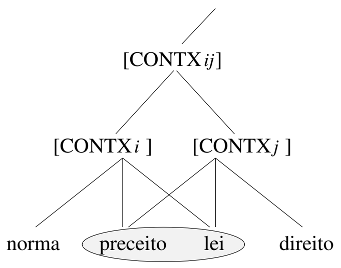


Fig. 3. Agglomerative clustering

Two basic classes are compared and then aggregated into a new more general class if they fulfil three specific conditions:

1. They must have the same number of words. We consider that two classes are compared in a more efficient manner when they have the same number of elements. Indeed, nonsensical results could be obtained if we compare large classes, which still remain polysemic and then heterogeneous, to the small classes that are included in them.
2. They must share $n-1$ words. Two classes sharing $n-1$ words are aggregated into a new class of $n+1$ members. Indeed, two classes with the same number of elements only differing in one word may be considered as semantically close.
3. They must have the highest weight. The weight of a class corresponds to the number of occurrences of the class as a subset of other classes (within $n+20$ supersets). Intuitively, the more a class is included in larger classes, the more semantically homogeneous it should be. Only those classes with the highest weight will be compared and aggregated.

Note that clustering does not rely here on a statistical distance between classes. Rather, clustering is guided by a set of constraints, which have been empirically defined considering linguistic data. Due to the nature of these constraints, the clustering process should start with small size classes with n elements, in order to create larger classes of $n + 1$ members. All classes of size n that fulfil the conditions stated above are aggregated into $n + 1$ clusters. In this agglomerative clustering strategy, level n is defined by the classes with n elements. The algorithm continues merging clusters at more complex levels and stops when there are no more clusters fulfilling the three conditions.

4.3 Tests and Evaluation

We used a small corpus with 1,643,579 word occurrences, selected from the case-law P.G.R. text corpora. First, the corpus was tagged by the part-of- speech tagger presented in [14]. Then, it was analysed in sequences of basic chunks by the partial parser presented in [18]. The chunks were attached using the right association heuristic so as to create binary dependencies. 211,976 different syntactic contexts with their associated word sets were extracted from these dependencies. Then, we filter these contextual word sets by using the method described above so as to obtain a list of basic classes.

In order to test our clustering strategy, we start the algorithm with basic classes of size 4 (i.e., classes with 4 elements). We have $7,571$ basic classes with 4 elements, but only a small part of them fills the clustering conditions so as to form $1,243$ clusters with 5 elements. At level 7, there are still 600 classes filling the clustering conditions, 263 at level 9, 112 at level 11, 38 at level 13, and finally only 1 at level 19. In table 2, we show some of the clusters generated by the algorithm at different intermediate levels.[5]

Note that some words may appear in different clusters. For instance, `cargo` (*task/post*) is associated with nouns referring to activities (e.g., `actividade,`

[5] In the left column, the first number represents the weight of the set, i.e., its occurrences as subset of larger supersets; the second number represents class cardinality.

Table 2. Some clusters at levels 6, 7, 8, 9, 10, and 11

0006 (06)	aludir citar enunciar indicar mencionar referir *allude cite enunciate indicate mention refer*
0009 (07)	considerar constituir criar definir determinar integrar referir *consider constitute create define determinate integrate refer*
0002 (07)	actividade atribuição cargo função funções tarefa trabalho *activity attribution position/task function functions task work*
0003 (08)	administração cargo categoria exercício função lugar regime serviço *administration post rank practice function place regime service*
0002 (09)	abono indemnização multa pensão propina remuneração renda sanção vencimento *bail compensation fine pension fee remuneration rent sanction salary*
0007 (10)	administração autoridade comissão conselho direcção estado governo ministro tribunal órgão *administration authority commission council direction state government minister tribunal organ*
0026 (11)	alínea artigo código decreto diploma disposição estatuto legislação lei norma regulamento *paragraph article code decret diploma disposition statute legislation law norm regulation*

trabalho, tarefa (*activity, work, task*)), as well as with nouns referring to the positions where those activities are produced (e.g., cargo, categoria, lugar (*post, rank, place*)). The sense of polysemic words is represented by the natural attribution of a word to various clusters.

Table 3. Some clusters generated at level 12

0002 (12)	administração associação autoridade comissão conselho direcção entidade estado governo ministro tribunal órgão *administration association authority commission council direction entity state government minister government tribunal organ*
0002 (12)	administração assembleia autoridade comissão conselho direcção director estado governo ministro tribunal órgão *administration assembly authority commission council direction director state government minister government tribunal organ*
0002 (12)	assembleia autoridade câmara comissão direcção estado europol governo ministério pessoa serviço órgão *assembly authority chamber commission direction state europol government state_department person service organ*
0002 (12)	administração autoridade comissão conselho direcção empresa estado gestão governo ministério serviço órgão *administration authority commission council direction firm state management gouvernment state_department person service organ*

Since this clustering strategy have been conceived to assure the semantic homogeneity of clusters, it does not really assure that each cluster represents an independent semantic class. Hence, two or more clusters can represent the same contextual-based class and, then, the same semantic restriction. Let's see

Table 3. Intuitively, the four clusters, which have been generated at level 12, refer to a general semantic class: *agentive_entities*. Yet, the algorithm is not able to aggregate them into a more general cluster at level 13, since they do not fill condition 2. Further work should refine the algorithm in order to solve such a problem.

Note that the algorithm does not generate ontological classes like *human beings, institutions, vegetables, dogs*,... but context-based semantic classes associated with syntactic contexts. Indeed, the generated clusters are not linguistic-independent objects but semantic restrictions taking part in the syntactic analysis of sentences. This way, the words `autoridade, pessoa, administração`, etc. (*authority, person, administration*) belong to the same contextual class because they share a great number of syntactic contexts, namely they appear as the subject of verbs such as `aprovar, revogar, considerar`, ... (*approve, repeal, consider*). Those nouns do not form an ontological class but rather a linguistic class used to constrain the syntactic word combination. More precisely, we may infer that the following syntactic contexts:

$$\begin{array}{l} [\lambda x^{\uparrow}(subj; aprovar^{\downarrow}, x^{\uparrow})] \\ [\lambda x^{\uparrow}(subj; revogar^{\downarrow}, x^{\uparrow})] \\ [\lambda x^{\uparrow}(subj; considerar^{\downarrow}, x^{\uparrow})] \end{array}$$

share the same selection restrictions since they are used to build a context-based semantic class constituted by words like `autoridade, pessoa, administração`, ... As has been said, the acquired selection restrictions should be used to check the attachment hypotheses concerning the candidate dependencies previously extracted.

5 Conclusion and Further Work

This paper has presented a particular unsupervised strategy to automatically learn context-based semantic classes used as restrictions on syntactic combinations. The strategy is mainly based on two linguistic assumptions: co-specification hypothesis, i.e., the two related expressions in a binary dependency impose semantic restrictions to each other, and contextual hypothesis, i.e., two syntactic contexts share the same semantic restrictions if they cooccur with the same words.

The system introduced in this paper will be applied to attachment resolution in syntactic analysis. The degree of efficacy in such a task may serve as a reliable evaluation for measuring the soundness of our learning strategy. This is part of our current work.

References

1. Roberto Basili, Maria Pazienza, and Paola Velardi. Hierarchical clustering of verbs. In *Workshop on Acquisition of Lexical Knowledge from Text*, pages 56–70, Ohio State University, USA, 1993.

2. Gilles Bisson, Claire Nédellec, and Dolores Canamero. Designing clustering methods for ontology building: The mo'k workbench. In *Internal rapport*, citerseer.nj.nec.com/316335.html, 2000.
3. Ido Dagan, Lillian Lee, and Fernando Pereira. Similarity-based methods of word coocurrence probabilities. *Machine Learning*, 43, 1998.
4. David Faure and Claire Nédellec. Asium: Learning subcategorization frames and restrictions of selection. In *ECML98, Workshop on Text Mining*, 1998.
5. David Faure. *Conception de méthode d'aprentissage symbolique et automatique pour l'acquisition de cadres de sous-catégorisation de verbes et de connaissances sémantiques à partir de textes : le système ASIUM.* PhD thesis, Université Paris XI Orsay, Paris, France, 200.
6. Francesc Ribas Framis. On learning more appropriate selectional restrictions. In *Proceedings of the 7th Conference of the European Chapter of the Association for Computational Linguistics*, Dublin, 1995.
7. Pablo Gamallo, Caroline Gasperin, Alexandre Agustini, and Gabriel P. Lopes. Syntactic-based methods for measuring word similarity. In *Text, Speech, and Discourse (TSD-2001).* Berlin:Springer Verlag (to appear), 2001.
8. Pablo Gamallo. *Construction conceptuelle d'expressions complexes: traitement de la combinaison nom-adjectif.* PhD thesis, Université Blaise Pascal, Clermont-Ferrand, France, 1998.
9. Gregory Grefenstette. *Explorations in Automatic Thesaurus Discovery.* Kluwer Academic Publishers, USA, 1994.
10. Gregory Grefenstette. Evaluation techniques for automatic semantic extraction: Comparing syntatic and window based approaches. In Branimir Boguraev and James Pustejovsky, editors, *Corpus processing for Lexical Acquisition*, pages 205–216. The MIT Press, 1995.
11. Ralph Grishman and John Sterling. Generalizing automatically generated selectional patterns. In *Proceedings of the 15th International on Computational Linguistics (COLING-94)*, 1994.
12. D. Hindle. Noun classification form predicate-argument structures. In *Proceedings of the 28th Meeting of the ACL*, pages 268–275, 1990.
13. Dekang Lin. Automatic retrieval and clustering of similar words. In *COLING-ACL'98*, Montreal, 1998.
14. Nuno Marques. *Uma Metodologia para a Modelação Estatística da Subcategorização Verbal.* PhD thesis, Universidade Nova de Lisboa, Lisboa, Portugal, 2000.
15. Fernando Pereira, Naftali Tishby, and Lillian Lee. Distributional clustering of english words. In *Proceedings of the 30th Annual Meeting of the Association of Comptutational Linguistics*, pages 183–190, Columbos, Ohio, 1993.
16. James Pustejovsky. *The Generative Lexicon.* MIT Press, Cambridge, 1995.
17. Philip Resnik. Semantic similarity in taxonomy: An information-based measure and its application to problems of ambiguity in natural language. *Journal of Artificial Intelligence Research*, 11:95–130, 1999.
18. V. Rocio, E. de la Clergerie, and J.G.P. Lopes. Tabulation for multi-purpose partial parsing. *Journal of Grammars*, 4(1), 2001.
19. Satoshi Sekine, Jeremy Carrol, Sofia Ananiadou, and Jun'ichi Tsujii. Automatic learning for semantic collocation. In *Proceedings of the 3rd Conference on Applied Natural Language Processing*, pages 104–110, 1992.
20. Tokunaga Takenobu, Iwayama Makoto, and Tanaka Hozumi. Automatic thesaurus construction based on grammatical relations. In *Proceedings of IJCAI-95*, 1995.
21. Luis Talavera and Javier Béjar. Integrating declarative knowledge in hierarchical clustering tasks. In *Intelligent Data Analysis*, pages 211–222, 1999.

Classification Rule Learning with APRIORI-C

Viktor Jovanoski and Nada Lavrač

J. Stefan Institute, Jamova 39, 1000 Ljubljana, Slovenia
{Viktor.Jovanoski, Nada.Lavrac}@ijs.si

Abstract. This paper presents the APRIORI-C algorithm, modifying the association rule learner APRIORI to learn classification rules. The algorithm achieves decreased time and space complexity, while still performing exhaustive search of the rule space. Other APRIORI-C improvements include feature subset selection and rule post-processing, leading to increased understandability of rules and increased accuracy in domains with unbalanced class distributions. In comparison with learners which use the covering approach, APRIORI-C is better suited for knowledge discovery since each APRIORI-C rule has high support and confidence.

1 Introduction

Mining of association rules has received a lot of attention in recent years. Compared to other machine learning techniques, its main advantage is a low number of database passes done when searching the hypothesis space, whereas its main disadvantage is time complexity.

One of the best known association rule learning algorithms is APRIORI [3,4]. This algorithm was extensively studied, adapted to other areas of machine learning and data mining, and successfully applied in many problem domains [5,15,1,14,2].

An association rule R has the form $X \Rightarrow Y, \quad for\ X, Y \subseteq I$, where I is a set of all items[1], and X and Y are *itemsets*. If $freq(X)$ denotes the number of transactions that are supersets of itemset X, and N the number of all transactions, then $Support(X) = \frac{freq(X)}{N}$. Each rule is associated with its *confidence* and *support*: $Confidence(R) = \frac{freq(X \cup Y)}{freq(X)}$, and $Support(R) = \frac{freq(X \cup Y)}{N}$.

This paper presents our algorithm APRIORI-C, based on APRIORI, whose modifications enable it to be used for classification. The basic APRIORI-C algorithm is described in Section 2. Given that the idea of using association rules for classification is not new [13], the main contribution of this paper are substantially decreased memory consumption and time complexity (described in Section 2), further decreased time-complexity by feature subset selection (Section 3), and improved understandability of results by rule post-processing (Section 4).

[1] In machine learning terminology, an item is a binary feature. In association rule learning, a binary attribute $A_i = v_j$ is generated for each value v_j of a discrete attribute A_j. For numeric attributes, items are formed by attribute discretization. Throughout this paper, items and features are used as synonyms.

2 The Core of APRIORI-C

Association rule learning can be adapted for classification purposes by implementing the following steps:

1. Discretize continuous attributes.
2. For each discrete attribute with N values create N items: for a discrete value v_i, i-th item of this attribute is assigned value 1, others are assigned value 0. In the case of a missing value, all items are assigned value 0.
3. Run an association rule learning algorithm.
4. Collect rules whose right-hand side consist of a single item, representing a value of the target attribute.
5. Use this set of rules to classify unclassified examples.

To classify an unclassified example with all the rules found by the algorithm, first, sort rules according to some criteria, next, go through the list of all rules until the first rule that covers the example is found, and finally, classify the example according to the class at the right-hand side of this rule. If no rule covers the example, mark the example as unclassified.

The above modifications are straightforward. Nevertheless, to better adapt the algorithm to classification purposes, APRIORI-C includes the following optimizations:

Classification rule generation Rules with a single target item at the right-hand side can be created during the search. To do so, the algorithm needs to save only the supported itemsets of sizes k and $k+1$. This results in decreased memory consumption (improved by factor 10). Notice, however, that this does not improve the algorithm's time complexity.

Prune irrelevant rules Classification rule generation can be supressed if one of the existing generalizations of the rule has support and confidence above the given thresholds. To prevent rule generation, the algorithm simply excludes the corresponding itemset from the set of supported *k+1*-itemsets. Time and space complexity reduction are considerable (improved by factor 10 or more).

Prune irrelevant items If an item cannot be found in any of the itemsets containing the target item, then it is impossible to create a rule containing this item. Hence, APRIORI-C prunes the search by discarding all itemsets containing this item.

The above optimizations assume that we want to find only rules which have a single item, called a *target* item, at their right-hand side. Consequently, one has to run the algorithm for each target item, representing each individual class. The algorithm, named APRIORI-C, is outlined in Figure 1.

APRIORI-C was evaluated on 17 datasets from the UCI ML databases repository (*ftp://ftp.ics.uci.edu/pub/machine-learning-databases/*). Most datasets contain continuous attributes which had to be discretized. This was done by using the K-Means clustering algorithm [16,7], which was an arbitrary choice.

```
for each target item $T$
    $k = 1$
    $C_1$ = set of all 1-itemsets
    check the support of all itemsets in $C_1$
    delete unsupported itemsets from $C_1$
    while $C_k$ not empty do
            build all potentially supported k+1-itemsets
            put them into $C_{k+1}$
            check their support
            delete unsupported itemsets from $C_{k+1}$
            find all rules that have target item $T$ at their right-hand side
            delete all itemsets that were used for constructing rules
            detect irrelevant items and delete irrelevant itemsets
            $k = k + 1$
    end while
endfor
```

Fig. 1. The APRIORI-C algorithm.

In the experiments, *MinConfidence* was set to 0.9, *MinSupport* to 0.03 and *MaxRuleLength* to 5. This setting works well for most of the datasets. For few other datasets, *MinSupport* had to be set to a higher value, e.g., 0.05 or even 0.1, to prevent combinatorial explosion. In contrast, for some other datasets the algorithm didn't find any rule, hence *MinSupport* was set to 0.01. Using 10-fold cross-validation, the accuracy of APRIORI-C was compared to the accuracies of other machine learning algorithms (results reported in [18], using default parameter settings; notice that the accuracy of these algorithms could be further improved by parameter tuning). Results in Table 1 show that in datasets with few or no continuous attributes, APRIORI-C is at least as good as other algorithms, sometimes even slightly better (*glass*, *tic-tac-toe* and *vote*). However, APRIORI-C performs poorly on the datasets containing continuous attributes (e.g., *balance*, *waveform* and *wine*). Except for APRIORI-C, all the other algorithms handle continuous attributes by themselves and not in pre-processing. The poor results can be mainly attributed to the suboptimal discretization currently used in APRIORI-C.

3 Pre-processing by Feature Subset Selection

The most serious problem of APRIORI-C is that increased number of features results in an exponential increase in time complexity. Attribute correlations can also seriously affect the performance and accuracy. To deal with these problems, *feature subset selection* was employed to select a feature subset, sufficient for the learner to construct a classifier of similar accuracy in shorter time. The following four approaches were tested: statistical correlation, odds ratio, and two variants of the *RELIEF* algorithm [11,17,12]. Experiments showed that in the association rule learning context, feature subset selection needs to be sensitive to the difference in the meaning of attribute values 0 and 1: while value 1 is favorable for rule construction, value 0 is not used in a constructed rule, it just prohibits rule construction.

Table 1. Results of APRIORI-C compared to other algorithms.

Dataset	C4.5	CN2	kNN	Ltree	LBayes	APRIORI-C
australian	0.85	0.83	0.85	**0.87**	0.86	0.86
balance	0.77	0.81	0.80	**0.93**	0.87	0.72
breast-w	0.94	0.94	0.95	0.94	**0.97**	0.92
bridges-td	0.85	0.85	0.85	0.85	**0.88**	0.83
car	0.92	**0.95**	0.94	0.88	0.86	0.85
diabetes	0.74	0.74	0.73	0.75	**0.77**	0.72
echocardiogram	0.65	0.66	0.69	0.64	**0.71**	0.65
german	0.72	0.73	0.70	0.73	**0.75**	0.70
glass	0.70	0.65	0.70	0.68	0.63	**0.77**
hepatitis	0.79	0.80	**0.85**	0.82	**0.85**	0.80
hypothyroid	**0.99**	**0.99**	0.97	**0.99**	0.96	0.95
image	**0.97**	0.86	**0.97**	**0.97**	0.92	0.77
iris	0.95	0.93	0.95	0.97	**0.98**	0.96
tic-tac-toe	0.85	0.98	0.91	0.82	0.70	**0.99**
vote	**0.96**	0.95	0.90	0.95	0.9	**0.96**
waveform	0.76	0.69	0.81	0.85	**0.86**	0.34
wine	0.94	0.93	0.97	0.97	**0.99**	0.88

Statistical Correlation In association rule learning, strong correlations among items cause drastic increase of the number of supported itemsets. Correlated items were removed by first testing all the pairs of non-target items and removing one of them if the correlation was above the user defined threshold $MaxCorr$. The algorithm complexity is $\mathbf{O}(n^2)$ w.r.t. the number of items.

Odds Ratio For target class C_1, union of all the non-target classes C_2, and n training examples, *Odds ratio* is computed for each feature F as follows: $OddsRatio(F) = log\frac{odds(F|C_1)}{odds(F|C_2)}$, where $odds(F)$ equals $\frac{\frac{1}{n^2}}{1-\frac{1}{n^2}}$ if $p(F) = 0$; equals $\frac{1-\frac{1}{n^2}}{\frac{1}{n^2}}$ if $p(F) = 1$; and finally, equals $\frac{p(F)}{1-p(F)}$ if $p(F) \neq 0$ and $p(F) \neq 1$. Probabilities are estimated using the relative frequency. Features are then sorted in the descending order of odds ratio values and the number of items is selected w.r.t the *RatioOfSelectedItems* parameter. Given that *Odds ratio* distinguishes between attribute values 1 and 0, it performs very well (see also results in [17]).

VRELIEF in VRELIEF2 In the *RELIEF* algorithm below [12], q is a list of quality estimations, $SampleSize$ is the user-defined number of randomly selected examples, and $d_{rx,i}$ is the distance between examples r and x for item I_i.

```
for each item $I_i$ do $q[I_i] = 0$
for j = 1 to Sample size do
    randomly select an example r
    find nearest hit t and nearest miss s
    for each item $I_i$ do
        $q[I_i] := q[I_i] + \frac{d_{rs,i} - d_{rt,i}}{Samplesize}$
sort items according to measure q
select the best items
```

RELIEF is an excellent algorithm for feature subset selection, but in our problem it performs poorly since it makes no distinction between attribute values 0 and 1. Consequently, the original RELIEF distance function was changed: *VRELIEF* assigns higher quality to an item with value 1 for examples of the same class. Despite the significant accuracy increase, the algorithm often selects items that cause unnecessary increase in time complexity. Hence, the distance function in *VRELIEF2* was changed so that items that increase the complexity without increasing accuracy receive lower quality estimation. The distance computations for training examples, labeled a and b, and values of the tested items for these two examples, are shown below:

Class		**RELIEF**				**VRELIEF**				**VRELIEF2**			
membership		**value$_a$/value$_b$**				**value$_a$/value$_b$**				**value$_a$/value$_b$**			
		1/1	**0/1**	**1/0**	**0/0**	**1/1**	**0/1**	**1/0**	**0/0**	**1/1**	**0/1**	**1/0**	**0/0**
Class(a)=+	Class(b)=+	0	1	1	0	1	0	0	-1	2	0	0	-2
Class(a)=+	Class(b)=-	0	1	1	0	0	-1	1	0	-1	-2	2	0
Class(a)=-	Class(b)=+	0	1	1	0	0	1	-1	0	-1	2	-2	0
Class(a)=-	Class(b)=-	0	1	1	0	-1	0	0	1	-2	-1	-1	2

Results Algorithms *Odds Ratio*, *VRELIEF* and *VRELIEF2* were tested with values of parameter *RatioOfSelectedItems* set to 0.2, 0.4, 0.6 and 0.8. Algorithm *Statistical Correlation* was tested with values of parameter $MaxCorr$ set to 0.2, 0.4, 0.6 and 0.8. For each value of the parameter, 10-fold cross-validation was performed and statistical t-test computed to see whether the best algorithm is significantly better than the others, both concerning the accuracy and complexity. Below is the summary of results:

– The *Odds Ratio* algorithm is a good choice for nearly every value of the *RatioOfSelectedItems* parameter. Other algorithms often come close, but never surpass it. The values around 0.4 of this parameter usually suffice for achieving the same accuracy as if no feature subset selection were performed.
– The *Statistical correlation* algorithm is often useful for cleaning the dataset as it only reduces the time complexity and retains the quality of results. The values around 0.5 of the $MaxCorr$ parameter are usually sufficient to drastically decrease the time complexity while retaining the same accuracy. Notice that this algorithm can easily be used in combination with other algorithms.
– The *VRELIEF* and *VRELIEF2* algorithms produce worse results in terms of accuracy when compared to *Odds Ratio* but they are better at decreasing the time complexity. The difference in performance between *VRELIEF* and *VRELIEF2* is statistically insignificant.

4 Post-processing by Rule Subset Selection

Rule redundancy It is common that many subsets of the set of constructed rules cover almost the same training examples.

Rule overflow The induced classifier is often not understandable since it includes too many rules. The user's upper limit of understanding is a classifier with 10–15 rules. Depending on the choice of parameters, the original algorithm often created classifiers with more than 100 rules (even more than 500).

Limited rule coverage The classifier often gives no answer, since no rule in the classifier fires for an instance to be classified. This problem is very serious when dealing with highly unbalanced class distributions.

Majority class domination When class distribution is unbalanced, the rules classifying to the majority class prevail: there are so many of those, that rules for the minority class hardly make it into the classifier. This causes poor accuracy when classifying instances of the minority class.

To deal with the above problems, APRIORI-C performs rule post-processing to select a rule subset with a comparable classification accuracy.

The *default* Rule Assignment After the algorithm selects the rules to be used in a classifier, some unclassified examples will probably remain in the set of training examples L. These unclassified examples contribute the majority of wrong classifications. Previous research confirmed that the probability of a wrong classification is much lower than the probability of the disability to classify an example. The simplest approach is to add a rule at the end of the rule list that will unconditionally classify an example to the majority class of the unclassified examples. The same approach, solving the problem of limited rule coverage, is used in the CN2 algorithm [8].

Select N Best Rules (and Select N Best Rules for Each Class) This algorithm uses the covering approach for rule subset selection. The number of rules to be selected is specified by the user-defined value of parameter N. The algorithm first selects the best rule (with highest support), then eliminates all the covered examples, sorts the remaining rules and again selects the best rule. This is repeated until the number of selected rules reaches the value of the parameter N or until there are no more rules to select or until there are no more examples to cover. The algorithm returns a sorted list of rules, and adds the default rule.

```
select $N$
$C = \{\}$
put all rules into $T$
put all the examples into $L$
sort the rules in $T$ according to their support and confidence in $L$
while $|C| < N$ and $|L| > 0$ and $|T| > 0$ do
    remove best rule $r$ from $T$ and add it to $C$
    remove the examples, covered by rule $r$, from $L$
    recalculate support and confidence of rules in T
end while
build the default rule and add it to $C$
output = $C$
```

This algorithm solves the problem of rule redundancy and rule overflow. Clearly, it also gives classifiers that execute faster. But it remains vulnerable to the problem of unbalanced class distribution. To solve this problem, the algorithm can be modified to **select N rules for each class** (if so many rules exist for each class). In this way the rules for the minority class also find their way into the constructed classifier.

Results Each algorithm was tested with several values of parameter N: 1, 2, 5, 10, 15 and 20. The main observations, confirmed by statistical tests, are outlined below:

- Algorithms USE N BEST RULES and USE N BEST RULES FOR EACH CLASS do not differ in accuracy. Only when there are significant differences in class distributions, the USE N BEST RULES FOR EACH CLASS algorithm performs better.

Next, both algorithms increase their accuracy significantly when using the *default* rule. This increase gets smaller with increasing value of parameter N, but it still remains noticeable. Finally, both algorithms achieve good results in terms of accuracy and understandability when the value of parameter N reaches 10. Accuracy results are comparable to the original algorithm.
- We have implemented and tested also the CONFIRMATION RULE SUBSET SELECTION algorithm [9] which also employs a greedy approach, but without removing the covered examples. The accuracy of this algorithm is always slightly lower than the accuracy of the other two algorithms, since this algorithm does not build the *default* rule.
- When comparing classifiers without the *default* rule, all algorithms achieve similar performance. This is probably due to the good quality of the rule set given as input to all the algorithms.

5 Conclusions

Using association rules for classification purposes has been addressed also by other researchers. Liu et al. [13] address the same problem. Their approach is similar to ours, but their algorithm for selecting rules to generate a classifier is more complicated and slower. They perform no feature subset selection and no post-processing. Bayardo and Agrawal [5,6] address the problem of mining constraint association rules. Implementing an efficient search for association rules given item constraints (useful for interactive mining) is described by Goethals and Bussche [10]. Their method of pruning is closely related to the APRIORI-C method of eliminating irrelevant itemsets.

This paper presents a new algorithm from the field of association rule learning that successfully solves classification problems. The APRIORI-C algorithm is a modification of the well-known APRIORI algorithm, making it more suitable for building rules for classification, due to the decreased time and space complexity, whereas the algorithm still exhaustively searches the rule space. Feature subset selection is best done using *Odds Ratio*, whereas post-processing is best done using the SELECT N BEST RULES or the SELECT N BEST RULES FOR EACH CLASS algorithms (for $N = 10$), together with the *default* rule.

In UCI domains APRIORI-C achieved results comparable to algorithms that have been designed for classification purposes. Given comparable accuracy, the advantage of APRIORI-C is that each of its rules represents a reasonable “chunk” of knowledge about the problem. Namely, each individual rule is guaranteed to have high support and confidence. This is important for knowledge discovery, as shown in the CoIL challenge: despite average classification accuracy, the induced rules were evaluated as very useful for practice. Most of the practically useful rules (or, better yet, conclusions) reported in the CoIL challenge were discovered by APRIORI-C as well.

Acknowledgements The work was supported by the Slovenian Ministry of Education, Science and Sport, and the IST-1999-11495 project Data Mining and Decision Support for Business Competitiveness: A European Virtual Enterprise.

References

1. K. Ali, S. Manganaris and R. Shrikant. Partial Classification using Association Rules. In Proceedings of the Third International Conference on Knowledge Discovery and Data Mining KDD 1997, Newport Beach, California.

2. R. Agrawal, J.Gehrke, D. Gunopulos and P. Raghavan. Authomatic Subspace Clustering of High Dimensional Data for Data Mining Applications. In Proceedings of ACM SIGMOD Conference on Management of Data, 1998.
3. R. Agrawal, T. Imielinski and R. Shrikant. Mining Association Rules between Sets of Items in Large Databases. In Proceedings of ACM SIGMOD Conference on Management of Data, Washington, D.C., 1993. pp. 207-216.
4. R. Agrawal and R. Srikant. Fast Algorithms for Mining Association Rules. In Proc. of the 20th Int'l Conference on Very Large Databases, Santiago, Chile, September 1994. pp. 207-216.
5. R. J. Bayardo Jr., R. Agrawal and D. Gunopulos. Constraint-Based Rule Mining in large, Dense Databases. In Proceedings of the 15th International Conference on Data Engineering, 1999. pp. 188-197.
6. R. J. Bayardo Jr. and R. Agrawal. Mining the most Interesting Rules. In Proceedings of the 5th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 1999. pp. 145-154.
7. P. S. Bradley, U. M. Fayyad. Refining Initial Points for K-Means Clustering. In Proceedings of International Conference on Machine Learning ICML'98. pp. 91-99.
8. P. Clark and T. Niblett. The CN2 induction algorithm. Machine Learning 3: 261-283, 1989.
9. D. Gamberger, N. Lavrač, C. Groselj. Diagnostic Rules of Increased Reliability for Critical Medical Applications. In Proceedings of Artificial Intelligence in Medicine. Joint European Conference on Artificial Intelligence in Medicine and Medical Decision Making, AIMDM'99, Aalborg, Denmark, June 1999.
10. B. Goethals and J. v.d.Bussche. On Supporting Interactive Association Rule Mining. In Proceedings of the Second International Conference on Data Warehousing and Knowledge Discovery, Lecture Notes in Computer Science Springer-Verlag, September 4-6, 2000, London-Greenwich, United Kingdom.
11. D. Koller, M. Sahami. Toward Optimal Feature Selection. In ICML-96: Proceedings of the Thirteenth International Conference on Machine Learning, pp. 284-292, San Francisco, CA: Morgan Kaufmann.
12. I. Kononenko. On Biases in Estimating Multi-Valued Attributes. In Proceedings of the 14th International Joint Conference on Artificial Intelligence IJCAI-95, pp. 1034-1040, 1995.
13. B. Liu, W. Hsu and Y. Ma. Integrating Classification and Association Rule Mining. Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining KDD'98, New York, USA, 1998.
14. H. Mannila and H. Toivonen. Discovering Generalized Episodes using Minimal Occurences, In Proceedings of the Second International Conference on Knowledge Discovery and Data Mining KDD96, 1996.
15. N. Megiddo and R. Shrikant. Discovering Predictive Association Rules. In Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining KDD'98, New York, USA, 1998.
16. M. Meila, D. Heckerman. An Experimantal Comparison of Several Clustering and Initialisation Methods. Technical report MSR-TR-98-06, Microsoft research, 1998.
17. D. Mladenić. Feature Subset Selection in Text-Learning. In Proceedings of 10th European Conference on Machine Learning ECML'98. pp. 121-129.
18. L. Todorovski and S. Džeroski. Combining Multiple Models with Meta Decision Trees. In Proc. ECML2000 Workshop on Meta-learning: Building Automatic Advice Strategies for Model Selection.

Proportional Membership in Fuzzy Clustering as a Model of Ideal Types

S. Nascimento[1,2], B. Mirkin[3], and F. Moura-Pires[4]

[1] Faculdade de Psicologia e Ciências da Educação–LEAD, UL
[2] Dep. Informática and CENTRIA, FCT-UNL
PORTUGAL
[3] School of Computer Science and Information Systems, Birkbeck College
London, UK
[4] Departamento de Informática, Universidade de Évora
PORTUGAL

Abstract. The goal of this paper is to further investigate the extreme behaviour of the fuzzy clustering proportional membership model (FCPM) in contrast to the central tendency of fuzzy *c*-means (FCM). A data set from the field of psychiatry has been used for the experimental study, where the cluster prototypes are indeed extreme, expressing the concept of 'ideal type'. While augmenting the original data set with patients bearing less severe syndromes, it is shown that the prototypes found by FCM are changed towards the more moderate characteristics of the data, in contrast with the almost unchanged prototypes found by FCPM, highlighting its suitability to model the concept of 'ideal type'.

Keywords: Fuzzy clustering; fuzzy model identification; proportional membership; ideal type.

1 Introduction

In previous work [1], [2], the fuzzy clustering proportional membership model (FCPM) was introduced and studied. In contrast to other fuzzy clustering approaches, FCPM is based on the assumption that the cluster structure is reflected in the data from which it has been constructed. More specifically, the membership of an entity in a cluster expresses the proportion of the cluster prototype reflected in the entity, according to FCPM. One of the phenomena observed in an extensive experimental study performed with simulated data, was that the cluster prototypes tend to express an extreme rather than average behaviour within a cluster [2]. The other approaches such as fuzzy *c*-means (FCM) tend to produce clusters whose prototypes are averaged versions of the relevant entities (up to degrees of the relevance) [3]-[5].

The goal of this paper is to further investigate this property of FCPM. Having a cluster structure revealed in a data set, we question how sensible is this cluster structure with regard to augmenting of the data set by entities that bear more or less similarity to the cluster prototypes. We take a specific data set [6] from the field of psychiatry where the cluster prototypes are indeed extreme to express severe syndromes of mental conditions and we augment this set with patients bearing less severe syndromes and explore whether the prototypes are changed or not. Yes, they are changed towards the more moderate characteristics with FCM. However, there is almost no change under FCPM, which shows that this domain may be a natural habitat of the FCPM approach.

2 The Fuzzy Clustering Proportional Membership Model (FCPM)

2.1 The Proportional Membership Model

In the clustering model, we assume the existence of some prototypes, offered by the knowledge domain, that serve as "ideal" patterns to data entities. To relate the prototypes to observations, we assume that the observed entities share parts of the prototypes. The underlying structure of this model can be described by a fuzzy c-partition defined in such a way that the membership of an entity to a cluster expresses *proportion* of the cluster's prototype reflected in the entity (*proportional membership*). This way, the underlying structure is substantiated in the fitting of data to the "ideal" patterns.

To be more specific, let $X = [x_{kh}]$ denote a $n \times p$ entity-to-feature data table where each entity, described by p features, is defined by the row-vector $\mathbf{x}_k = [x_{kh}] \in \Re^p$ ($k = 1 \cdots n$; $h = 1 \cdots p$). Let data matrix X be preprocessed into $Y = [y_{kh}]$ by shifting the origin to the gravity center of all the entities (rows) in Y and rescaling features (columns) by their ranges. This data set can be structured according to a fuzzy c-partition which is a set of c clusters, any cluster i ($i = 1, \cdots, c$) being defined by: 1) its prototype, a row-vector $\mathbf{v}_i = [v_{ih}] \in \Re^p$, and 2) its membership values $\{u_{ik}\}$ ($k = 1 \cdots n$), so that the following constraints hold for all i and k:

$$0 \leq u_{ik} \leq 1 \tag{1a}$$

$$\sum_{i=1}^{c} u_{ik} = 1. \tag{1b}$$

Let us assume that each entity $\mathbf{y}_k$ of Y is related to each prototype $\mathbf{v}_i$ up to its membership degree u_{ik}; that is, u_{ik} expresses the proportion of $\mathbf{v}_i$ which is present in $\mathbf{y}_k$ in such a way that approximately $y_{kh} = u_{ik}v_{ih}$ for every feature h. More formally, we suppose that

$$y_{kh} = u_{ik}v_{ih} + \varepsilon_{ikh}, \tag{2}$$

where the residual values ε_{ikh} are as small as possible.

These equations provide a feedback from a cluster structure to the data.

According to (2), a clustering criterion can be defined to fit each data point to each of the prototypes up to the degree of membership. This goal is achieved by minimizing all the residual values ε_{ikh} via one of the least-squares criteria

$$E_m(U,V) = \sum_{i=1}^{c}\sum_{k=1}^{n}\sum_{h=1}^{p} u_{ik}^m (y_{kh} - u_{ik}v_{ih})^2, \tag{3}$$

subject to constraints (1a) and (1b), where $m = 0, 1, 2$ is a pre-specified parameter.

The equations in (2) along with the least-squares criteria (3) to be minimized by unknown parameters U and $V = (\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_c) \in \Re^{cp}$ for Y given, are designated as the generic FCPM model. The models corresponding to $m = 0, 1, 2$ are denoted as FCPM-0, FCPM-1, and FCPM-2, respectively [2].

In FCPM, both, prototypes and memberships, are reflected in the model of data generation. The equations (2) can be considered as a device to reconstruct the data from the model. The clustering criteria follow the least-squares framework to warrant that the reconstruction is as exact as possible. Other scalarizations of the idea of minimization of the residuals can be considered as well.

2.2 Ideal Type and FCPM

Each prototype, $\mathbf{v}_i$, according to (2), is an "ideal" point such that any entity, $\mathbf{y}_k$, bears a proportion of it, u_{ik}, up to the residuals. The proportion, u_{ik}, is considered as the value of membership of $\mathbf{y}_k$ to the cluster i, representing thus, the extent entity $\mathbf{y}_k$ is characterized by the corresponding "ideal" point $\mathbf{v}_i$. This allows us to consider this model as relevant to the concept of ideal type in logics. M. Weber [1864-1920] defines that 'The ideal type is such a combination of characteristics that no real entity can satisfy all of them, though the entities can be compared by their proximity to the ideal type' (quoted from [6]). An ideal type represents a concept self-contained and entirely separated from all the other (ideal) types.

As our simulation experiments have shown, the model indeed tends to reveal somewhat extreme points as the cluster prototypes, especially when FCPM-2 is employed [2]. This will be further explored in the remainder.

2.3 FCPM and FCM

The criteria (3) can be expressed in terms of the distances between observed entities $\mathbf{y}_k$ and cluster prototypes $\mathbf{v}_i$ as follows:

$$E_m(U,V) = \sum_{k=1}^{n}\sum_{i=1}^{c} u_{ik}^m d(\mathbf{y}_k, u_{ik}\mathbf{v}_i) \tag{4}$$

where $d(\mathbf{a}, \mathbf{b})$ is the Euclidean distance (squared) between p-dimensional vectors $\mathbf{a}$ and $\mathbf{b}$.

This shows that these criteria are parallel, to an extent, to those minimized in the popular fuzzy c-means method (FCM) [3]:

$$B_m(U,V) = \sum_{k=1}^{n}\sum_{i=1}^{c} u_{ik}^m d(\mathbf{y}_k, \mathbf{v}_i). \tag{5}$$

The major difference is that FCPM takes the distances between entities $\mathbf{y}_k$ and their 'projections' on the axes joining 0 and cluster prototypes while FCM takes the distances between $\mathbf{y}_k$ and prototypes themselves. This leads to the following differences in respective algorithms and solutions:

1. The alternating optimization algorithm is straightforward in FCM because its criterion (5) very well separates the sets of variables related to centroids and to membership. On the contrary, the alternating optimization in FCPM requires some work when optimizing the membership, because it is not separated from centroids in (4).
2. FCM gives poor results when $m = 0$ or $m = 1$ [3], while these values of m are admissible in FCPM and lead to interesting results [2]. In particular, in FCPM-0 each entity is considered as a part of each prototype, leading to such effects as automatically adjusting the number of clusters to smaller values. On the other hand, in FCPM-1 and FCPM-2 larger proportions u_{ik} have larger weights over the smaller ones, thus overcoming the rigidity of FCPM-0. We will stick to the most contrasting of these models, FCPM-2, in this paper.
3. FCM tends to produce prototypes as averaged versions of their clusters, while FCPM would tend keep prototypes outside of the data cloud.
4. In FCPM, the following phenomenon occurs which has no analogues in FCM. Given centroids $\mathbf{v}_i$, minimizing (4) over u_{ik} would tend to minimize distances $d(\mathbf{y}_k, u_{ik}\mathbf{v}_i)$ by projecting $\mathbf{y}_k$ onto axes joining 0 and $\mathbf{v}_i$ thus leading to $u_{ik}\mathbf{v}_i$ being these projections. This would require a careful choice of the location of the origin of the space in FCPM, which is irrelevant in FCM. In particular, putting the origin into the gravity center (grand mean) point would make all potential extreme prototypes much differing from each other in the framework of FCPM. Figure 1 illustrates this aspect geometrically.
   Consider two data points, $\mathbf{y}_1$ and $\mathbf{y}_2$, and a prototype, $\mathbf{v}_1$, in the original reference frame (Figure 1A). It is apparent from the figure that the projection of each data point onto prototype $\mathbf{v}_1$, $u_{11}\mathbf{v}_1$ and $u_{12}\mathbf{v}_1$ correspond to close values. Let us consider the reference frame centered on the mean of the data, $\overline{\mathbf{y}}$, instead (Figure 1B). Now, the prototype is $\mathbf{v}'_1$, and the projections, $u_{11}\mathbf{v}'_1$ and $u_{12}\mathbf{v}'_1$, of each data point, become much more distinct from each other. A similar construction may be devised for prototype $\mathbf{v}_2$ and $\mathbf{v}'_2$ in each reference frame. Moreover, if the original reference frame is used, one may question the utility of having two prototypes ($\mathbf{v}_1$ and $\mathbf{v}_2$), whereas in the transformed reference frame the corresponding prototypes ($\mathbf{v}'_1$ and $\mathbf{v}'_2$) become very much distinct.

2.4 FCPM Algorithm

An FCM like alternating optimization (*AO*) algorithm has been developed in order to minimize the FCPM criteria [2]. Each iteration of the algorithm consists of two steps as follows.

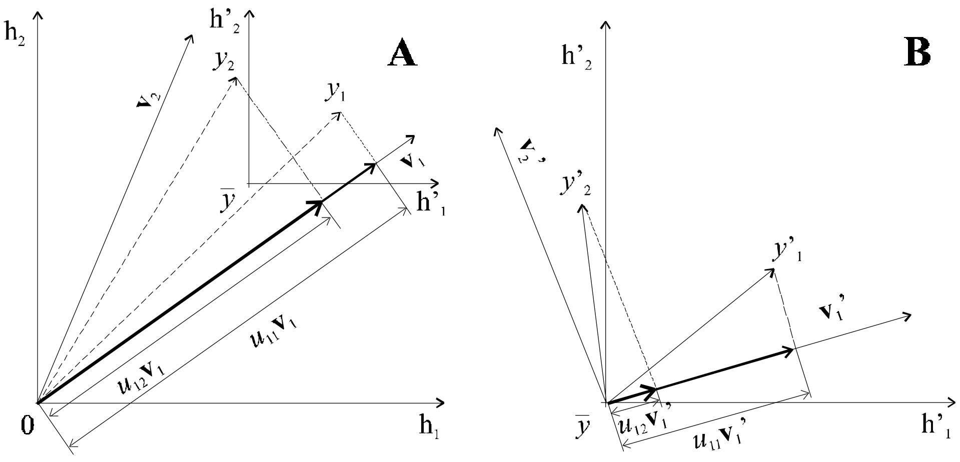


Fig. 1. How the distinction between prototypes $\mathbf{v}_i$ and corresponding projections, $\mathbf{u}_{ik}\mathbf{v}_i$, of data points $\mathbf{y}_k$ are affected by the choice of the origin of the space in the framework of FCPM model. **A.** Data points and prototypes in the original reference frame; **B.** Same data points and prototypes in the transformed reference frame, where its relative position to the original reference frame is depicted in **A.**

First, given a prototype matrix V, the optimal membership values are found by minimizing one of the criteria (3). In contrast to FCM, minimization of each criteria subject to constraints (1a) and (1b) is not an obvious task; it requires an iterative solution on its own. The gradient projection method [8],[9] has been selected for finding optimal membership values (given the prototypes). It can be proven that the method converges fast for FCPM-0 with a constant (anti)gradient stepsize.

Let us denote the set of membership vectors satisfying conditions (1a) and (1b) by Q. The calculations of the membership vectors $\mathbf{u}_k^{(t)} = \left[u_{ik}^{(t)}\right]$ are based on vectors $\mathbf{d}_k^{(t)} = \left[d_{ik}^{(t)}\right]$:

$$\begin{aligned} d_{ik}^{(t)} = u_{ik}^{(t-1)} - & \\ & 2\alpha_m \Big[(m+2)\left\langle v_i, v_i\right\rangle u_{ik}^{(m+1)} - \\ & 2(m+1)\left\langle v_i, y_k\right\rangle u_{ik}^{m} + m\left\langle y_k, y_k\right\rangle u_{ik}^{m-1}\Big] \end{aligned} \tag{6}$$

where α_m is a stepsize parameter of the gradient method. Then, $\mathbf{u}_k^{(t)}$ is to be taken as the projection of $\mathbf{d}_k^{(t)}$ in Q, denoted by $P_Q(\mathbf{d}_k^{(t)})$[1]. The process stops

[1] The projection $P_Q(\mathbf{d}_k^{(t)})$ is based on an algorithm we developed for projecting a vector $\mathbf{d}_k^{(t)}$ over the simplex of membership vectors $\mathbf{u}_k^{(t)}$; its description is omitted here.

when the condition $\left|U^{(t)} - U^{(t-1)}\right| \leq \varepsilon$ is fulfilled, with $|\cdot|_{err}$ an appropriate matrix norm.

Second, given a membership matrix U, the optimal prototypes are determined according to the first-degree optimum conditions as

$$v_{ih}^{(t)} = \frac{\sum_{k=1}^{n} \left(u_{ik}^{(t)}\right)^{m+1} y_{kh}}{\sum_{k=1}^{n} \left(u_{ik}^{(t)}\right)^{m+2}}, \tag{7}$$

with $m = 0, 1, 2$. Thus, the algorithm consists of “major” iterations of updating matrices U and V and “minor” iterations of recalculation of membership values in the gradient projection method within each of the “major” iterations. The algorithm starts with a set $V^{(0)}$ of c arbitrarily specified prototype points in $\Re^p$ and $U^{(0)}$; it stops when the difference between successive prototype matrices becomes small or a maximum number of iterations t_{1_max} is reached (in our experiments $t_{1\_max} = 100$).

The algorithm converges only locally (for FCPM-1 and FCPM-2). Moreover, with a “wrong” number of clusters pre-specified, FCPM-0 may not converge at all due to its peculiar behaviour: it may shift some prototypes to infinity [2].

3 Experimental Study

3.1 Contribution Weights

The concept of ‘importance weight’ of a variable is useful in cluster analysis. In particular the following relative contributions of a variable h to a cluster i, $w(h|i)$ and to the overall data scatter, $w(h)$, will be considered in this study [6], [7]:

$$w(h|i) = \frac{\overline{v}_{ih}^2}{\sum\limits_h \overline{v}_{ih}^2} \tag{8}$$

$$w(h) = \frac{\sum\limits_{i=1}^{c} \overline{v}_{ih}^2 n_i}{\sum\limits_{k,h} y_{kh}^2} \tag{9}$$

where $\overline{v}_{ih}$ denotes the gravity center of (hard) cluster i, i.e. $\overline{v}_{ih} = \sum_{k \in C_i} y_{kh}/n_i$, with $C_i = \{k : u_{ik} = 1\}$, and $n_i = |C_i|$, the number of entities belonging to cluster i. Note that the farther $\overline{v}_{ih}$ is from zero (which, due to our data standardization is the grand mean), the easier is to separate that cluster from the other ones in terms of variable h, which is reflected in the weight values. Due to this, $w(h|i)$ might be viewed as a measure of the “degree of interestingness” of variable h in cluster i with regard to its “standard” mean value.

Let us define the ‘most contributing’ features within a cluster and to the overall data structure, as the ones greater than the averages of (8) and (9), $\sum_h w(h|i)/p$ and $\sum_h w(h)/p$ respectively, which may act as frontier values.

Table 1. Relative contributions of the 17 psychosomatic variables (h_1-h_{17}) to classes D, M, S_s and S_p; and to the data scatter according to the original partition structure. The features values higher than corresponding averages are marked.

	D	M	S_s	S_p	General
h	$w(h\|D)$	$w(h\|M)$	$w(h\|S_s)$	$w(h\|S_p)$	$w(h)$
h_1	4.27	3.87	0.22	0.04	2.07
h_2	2.36	1.67	3.75	2.19	2.02
h_3	2.28	$\boxed{16.17}$	$\boxed{7.31}$	0.03	$\boxed{5.58}$
h_4	2.70	0.00	0.54	1.49	1.06
h_5	$\boxed{14.17}$	3.44	1.00	2.19	$\boxed{5.07}$
h_6	1.30	3.76	4.42	1.41	2.20
h_7	1.18	0.67	1.70	0.99	0.92
h_8	$\boxed{9.95}$	$\boxed{13.99}$	$\boxed{12.76}$	$\boxed{8.93}$	$\boxed{9.57}$
h_9	$\boxed{15.60}$	$\boxed{4.94}$	0.59	2.28	$\boxed{5.74}$
h_{10}	3.44	1.39	4.28	$\boxed{8.37}$	3.43
h_{11}	1.76	0.53	2.79	$\boxed{16.40}$	3.97
h_{12}	1.76	0.71	1.00	$\boxed{12.50}$	3.02
h_{13}	$\boxed{16.88}$	$\boxed{6.26}$	0.13	$\boxed{6.12}$	$\boxed{7.00}$
h_{14}	1.36	0.46	3.01	$\boxed{5.35}$	1.92
h_{15}	0.59	0.49	0.70	$\boxed{6.45}$	1.54
h_{16}	0.00	$\boxed{8.39}$	$\boxed{20.92}$	1.27	$\boxed{5.64}$
h_{17}	$\boxed{5.44}$	$\boxed{16.40}$	$\boxed{11.14}$	2.00	$\boxed{7.45}$
$\overline{w}$	5.00	4.89	4.50	4.60	4.01

3.2 Fuzzy Clustering of Mental Disorders Data

A mental disorders data set [6] was chosen, consisting of 44 patients, described by seventeen psychosomatic features (h_1-h_{17}). The features are measured on a severity rating scale taking values of 0 to 6. The patients are partitioned into four classes of mental disorders: depressed (D), manic (M), simple schizophrenic (S_s) and paranoid schizophrenic (S_p). Each class contains eleven consecutive entities that are considered 'archetypal psychiatric patients' of that class.

The mental disorders data set shows several interesting properties. First, there is always a pattern of features (a subset of h_1-h_{17}) that take extreme values (either 0 or 6) and clearly distinguish each class. Better still, some of these features take opposite values among distinct classes. However, some feature values are shared by classes leading to overlaps. Given these characteristics, each disease is characterized by 'archetypal patients' that show a pattern of extreme psychosomatic feature values defining a syndrome.

Table 1 presents the relative weights, $w(h|i)$ (defined by (8)), of features h_1-h_{17} to classes D, M, S_s and S_p and to the data scatter, $w(h)$ (9), according to the original partition structure. The corresponding averages, $\overline{w}$, are listed at the bottom row. All values are in percent, and the values corresponding to the '*most contributing'* features are marked.

The algorithms FCPM-2 and FCM (with its parameter $m = 2$) have been run starting from the same initial points, setting the number of clusters to four (i.e. $c = 4$). Table 2 shows the prototypes found by FCM and FCPM-2, in the original data scale, where the marked values belong to the set of the most contributing weight features within a cluster. Comparing the results, it is clear that the feature-values of FCPM-2 prototypes are more extreme than corresponding ones obtained by FCM (which is in accordance with our previous studies [2]). Moreover, in the prototypes found by FCPM-2 some of the feature values move outside the feature scale. In fact, all these 'extreme' values belong to the set of features that contribute the most to the whole partition or, at least, to a cluster (Table 1). In particular, these features present the highest discriminating power to separate the corresponding cluster from the remaining ones.

This property of FCPM prototypes goes in line with suggestions in data mining that the more deviant a feature is from a standard (the grand mean, in this case), the more interesting it is.

Concerning the membership values found, both algorithms assign the highest belongingness of an entity to its original class, correctly clustering all entities to the corresponding class[2].

3.3 Clustering of Augmented Mental Disorders Data

In order to study FCPM-2 behaviour on revealing extreme prototypes against FCM based central prototypes, the original data set should be modified to get in less expressed cases. To achieve that, each class was augmented with six mid-scale patients and three light-scale patients. Each new patient, $\mathbf{x}_g = [x_{gh}]$, was generated from a randomly selected original patient, $\mathbf{x}_k = [x_{kh}]$, applying the transformation: $x_{gh} = round\,(s_F \cdot x_{kh}) + t$ (for all $h = h_1, \cdots, h_{17}$), with scale-factor $s_F = 0.6$ to obtain a mid-scale patient and $s_F = 0.3$ to obtain a light-scale patient. The shift parameter t is a random number between 0 or 1.

Table 3 shows the prototypes ($\mathbf{v}$) found in the original data set followed by corresponding ones ($\mathbf{v}'$) for the augmented data, for each algorithm, where the values of features with the highest weights are also marked.

Most of FCM prototypes extreme feature values (in $\mathbf{v}$'s) move towards intermediate feature values (in $\mathbf{v}'$'s), showing FCM tendency to reveal central prototypes. Contrastingly, in FCPM-2 prototypes, the most weighting features maintain their full-scale (i.e. extreme) values, reinforcing the 'extreme' nature of FCPM-2 prototypes, and consequently, their sensitivity to the most discriminating features, despite the presence of new mid- and light-scale patients.

[2] The only exception occurs for entity (21) from class M, which is assigned to class S_p; the same phenomenon is reported in [6] for other clustering algorithms as complete linkage, K-means and separate-and-conquer.

Table 2. Cluster prototypes ($\mathbf{v}_D$, $\mathbf{v}_M$, $\mathbf{v}_{Ss}$, $\mathbf{v}_{Sp}$), revealed by FCM and FCPM-2, presented in the original space. Values corresponding to the most contributing weight features are marked.

	FCM				*FCP M-2*			
h	v_D	v_M	v_{Ss}	v_{Sp}	v_D	v_M	v_{Ss}	v_{Sp}
h_1	4	1	3	3	6	0	2	4
h_2	4	2	2	4	5	1	0	5
h_3	5	$\boxed{0}$	5	3	6	$\boxed{-2}$	6	4
h_4	2	3	3	4	1	3	3	6
h_5	$\boxed{5}$	1	1	1	$\boxed{7}$	0	0	0
h_6	2	5	2	4	1	6	0	5
h_7	1	1	2	3	0	0	3	4
h_8	$\boxed{0}$	$\boxed{6}$	$\boxed{1}$	5	$\boxed{-1}$	$\boxed{7}$	$\boxed{-1}$	6
h_9	$\boxed{6}$	1	2	2	$\boxed{8}$	0	1	1
h_{10}	2	4	2	5	0	4	1	6
h_{11}	2	2	2	$\boxed{5}$	1	2	1	$\boxed{7}$
h_{12}	1	1	1	4	0	1	0	6
h_{13}	$\boxed{5}$	$\boxed{0}$	2	$\boxed{1}$	$\boxed{7}$	$\boxed{-1}$	3	$\boxed{-1}$
h_{14}	3	4	2	5	2	4	1	6
h_{15}	3	3	3	$\boxed{5}$	2	2	2	$\boxed{7}$
h_{16}	2	$\boxed{0}$	$\boxed{5}$	2	3	$\boxed{-1}$	$\boxed{7}$	1
h_{17}	$\boxed{1}$	$\boxed{6}$	$\boxed{0}$	4	$\boxed{-1}$	$\boxed{8}$	$\boxed{-1}$	5

4 Conclusion

Modelling the concept of 'ideal type' through the notion of 'proportional membership' appears to be a promising aspect to be explored in the FCPM model.

The results of the experimental study clearly outline the extreme behaviour of FCPM prototypes in contrast with the central tendency of FCM prototypes, particularly when the data set was augmented with new patients more or less similar to the cluster prototypes. Since the extreme feature values of FCPM correspond to the most contributing weights, the proportional membership has a discriminating power to separate clusters from each other. Moreover, this extreme behaviour appears to be compatible with the notion of "interestingness" in data mining, since it is sensitive to the feature-values that are farthest from the average.

In terms of applications, the concept of 'ideal type' fits to such domain areas as psychiatry or market research, where a prototype (such as mental disturbance or consumer profile, in the examples given) is well characterized by extreme conditions. Modeling such a concept through fuzzy clustering seems appealing and useful as a part of the decision making process in those application areas.

The extension of this preliminary experimental study using more vast data sets is fundamental to firmly establish the found results and better outline the model properties and its relevance in the application to several domain areas.

Table 3. Prototypes found by FCM and FCPM-2, in the original data set (v's) and in the augmented data set (v''s), characterizing each mental disorder: D, M, S_s and S_p.

	FCM								$FCPM$-2							
h	v_D	v'_D	v_M	v'_M	v_{Ss}	v'_{Ss}	v_{Sp}	v'_{Sp}	v_D	v'_D	v_M	v'_M	v_{Ss}	v'_{Ss}	v_{Sp}	v'_{Sp}
h_1	4	4	1	1	3	2	3	3	6	5	0	0	2	2	4	4
h_2	4	4	2	2	2	2	4	4	5	5	1	1	0	0	5	5
h_3	5	4	$\boxed{0}$	$\boxed{1}$	5	4	3	3	6	5	$\boxed{-2}$	$\boxed{-1}$	6	5	4	4
h_4	2	2	3	3	3	2	4	4	1	1	3	3	3	2	6	5
h_5	$\boxed{5}$	$\boxed{4}$	1	1	1	1	1	1	$\boxed{7}$	$\boxed{7}$	0	0	0	0	0	0
h_6	2	2	5	4	2	2	4	4	1	1	6	6	0	0	5	5
h_7	1	1	1	1	2	2	3	2	0	0	0	0	3	2	4	4
h_8	$\boxed{0}$	$\boxed{0}$	$\boxed{6}$	$\boxed{5}$	$\boxed{1}$	$\boxed{1}$	5	4	$\boxed{-1}$	$\boxed{-1}$	$\boxed{7}$	$\boxed{7}$	$\boxed{-1}$	$\boxed{-1}$	6	6
h_9	$\boxed{6}$	$\boxed{5}$	1	1	2	2	2	2	$\boxed{8}$	$\boxed{7}$	0	0	1	1	1	1
h_{10}	2	2	4	3	2	2	5	4	0	0	4	4	1	1	6	6
h_{11}	2	2	2	2	2	2	$\boxed{5}$	$\boxed{5}$	1	1	2	2	1	0	$\boxed{7}$	$\boxed{7}$
h_{12}	1	1	1	2	1	1	4	4	0	1	1	1	0	0	6	6
h_{13}	$\boxed{5}$	$\boxed{4}$	$\boxed{0}$	$\boxed{1}$	2	2	$\boxed{1}$	$\boxed{1}$	$\boxed{7}$	$\boxed{7}$	$\boxed{-1}$	$\boxed{-1}$	3	2	$\boxed{-1}$	$\boxed{0}$
h_{14}	3	2	4	4	2	2	5	5	2	2	4	4	1	1	6	6
h_{15}	3	3	3	3	3	2	$\boxed{5}$	$\boxed{5}$	2	2	2	3	2	1	$\boxed{7}$	$\boxed{7}$
h_{16}	2	2	$\boxed{0}$	$\boxed{1}$	$\boxed{5}$	$\boxed{3}$	2	1	3	3	$\boxed{-1}$	$\boxed{-1}$	$\boxed{7}$	$\boxed{6}$	1	1
h_{17}	$\boxed{1}$	$\boxed{1}$	$\boxed{6}$	$\boxed{5}$	$\boxed{0}$	$\boxed{1}$	4	4	$\boxed{-1}$	$\boxed{0}$	$\boxed{8}$	$\boxed{7}$	$\boxed{-1}$	$\boxed{-1}$	5	4

Acknowledgments One of the authors (S.N.) acknowledges FCT-Portugal for the Ph.D. grant between 1997-1999 (PRAXIS XXI).

References

1. Nascimento, S., Mirkin, B., Moura-Pires, F.: Multiple Prototypes Model for Fuzzy Clustering. In Kok, J., Hand, D. and Berthold, M., (eds.), Advances in Intelligent Data Analysis. Third International Symposium, (IDA'99), Lecture Notes in Computer Science, 1642, Springer-Verlag. (1999) 269–279
2. Nascimento, S., Mirkin, B., Moura-Pires, F.: A Fuzzy Clustering Model of Data with Proportional Membership. The 19th International Conference of the North American Fuzzy Information Processing Society (NAFIPS 2000). IEEE (2000) 261-266
3. Bezdek, J.: Pattern Recognition with Fuzzy Objective Function Algorithms. Plenum Press, New York (1981)
4. Bezdek, J., Keller, J., Krishnapuram, R., Pal, T.: Fuzzy Models and Algorithms for Pattern Recognition and Image Processing. Kluwer Academic Publishers (1999)
5. Höppner, F., Klawonn, F., Kruse, R., Runkler, T.: Fuzzy Cluster Analysis. John Wiley and Sons (1999)
6. Mirkin, B.: Mathematical Classification and Clustering. Kluwer Academic Publishers (1996)
7. Mirkin, B.: Concept Learning and Feature Selection Based on Square-Error Clustering. In Machine Learning, **25**(1) (1999) 25-40

8. Polyak, B.: Introduction to Optimization. Optimization Software, Inc., New York (1987)
9. Bertsekas, D.: Nonlinear Programming. Athena Scientific (1995)

Evolution of Cubic Spline Activation Functions for Artificial Neural Networks

Helmut A. Mayer and Roland Schwaiger

Department of Computer Science
University of Salzburg
A–5020 Salzburg, Austria
{helmut,rschwaig}@cosy.sbg.ac.at

Abstract. The most common (or even only) choice of activation functions (AFs) for multi–layer perceptrons (MLPs) widely used in research, engineering and business is the logistic function. Among the reasons for this popularity are its boundedness in the unit interval, the function's and its derivative's fast computability, and a number of amenable mathematical properties in the realm of approximation theory. However, considering the huge variety of problem domains MLPs are applied in, it is intriguing to suspect that specific problems call for specific activation functions. Biological neural networks with their enormous variety of neurons mastering a set of complex tasks may be considered to motivate this hypothesis. We present a number of experiments evolving structure and activation functions of generalized multi–layer perceptrons (GMLPs) using the parallel netGEN system to train the evolved architectures. For the evolution of activation functions we employ cubic splines and compare the evolved cubic spline ANNs with evolved sigmoid ANNs on synthetic classification problems which allow conclusions w.r.t. the shape of decision boundaries. Also, an interesting observation concerning Minsky's Paradox is reported.

1 Introduction

While many researchers have investigated a variety of methods to improve *Artificial Neural Network* (ANN) performance by optimizing training methods, learn parameters, or network structure, comparably few work has been done towards using activation functions other than the logistic function. It has been shown that a two–hidden–layer MLP with sigmoidal activation function can implement arbitrary convex decision boundaries [1]. Moreover, such an MLP is capable of forming an arbitrarily close approximation to any continous nonlinear mapping [2]. However, common to these theorems is that the number of neurons in the layers is at best bounded, but can be extremely large for practical purposes. E.g., for a one–hidden–layer MLP Barron (1993) derived a proportionality of the sum square error (SSE) to the number of neurons according to $SSE \sim \frac{1}{\sqrt{T}}$ with T being the number of neurons (for target functions having a *Fourier* representation) [3].

If we consider the simple example of a rectangular function, it becomes obvious that a rectangular activation function can approximate this target function much easier than a sigmoidal function.

1.1 Related Work

Liu and Yao (1996) evolved the structure of *Generalized Neural Networks* (GNN) with two different activation function types, namely, sigmoid and Gaussian basis function. Experiments with the *Heart Disease* data set from the UCI machine learning benchmark repository revealed small differences in classification error slightly favoring GNNs over ANNs solely using sigmoid or Gaussian basis activation function [4].

Vecci et al. (1998) introduced the *Adaptive Spline Neural Network* (ASNN), where activation functions are described by *Catmull–Rom* cubic splines. ANN error (including a regularization term) then is not only dependent on the weights, but also on the spline parameters which together with the weights are adapated by backpropagation learning. It has been reported that ASNNs yield a performance comparable to networks using the logistic AF, while using a smaller number of hidden neurons than the conventional network [5].

Sopena et al. (1999) presented a number of experiments (with widely–used benchmark problems) showing that multilayer feed–forward networks with a sine activation function learn two orders of magnitude faster while generalization capacity increases (compared to ANNs with logistic activation function) [6].

2 Evolution of ANN Structure and Activation Functions

The technical platform for the evolution of GMLPs is the netGEN system searching a problem–adapted ANN architecture by means of an *Evolutionary Algorithm* (EA), and training the evolved networks using the *Stuttgart Neural Network Simulator* (SNNS) [7]. In order to speed up the evolutionary process, an arbitrary number of workstations may be employed to train individual ANNs in parallel.

The ANN's genetic blueprint is based on a direct encoding suggested by Miller et al. [8]. The basic structure of the genotype is depicted in Figure 1.

Learn Parameters	Neuron Parameters	Structure Parameters

Fig. 1. The organization of the ANN genotype.

The binary encoded GMLP chromosome contains three main sections. The *Learn Parameters* encode values used for ANN training, the *Neuron Parameters* are used to describe the activation function of each neuron, and the *Structure Parameters* explicitely specify each connection of the network (direct encoding).

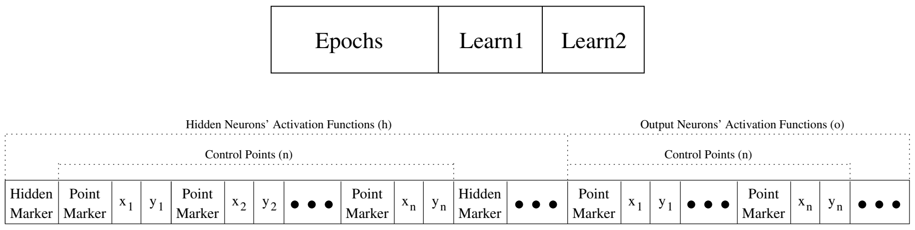


Fig. 2. Details of the learn parameter section (above) and the neuron parameter section (below) of the ANN genotype.

Figure 2 shows the details of the learn parameter and the neuron parameter section.

The learn parameter section contains the number of epochs and two learning parameters for the selected training algorithm *Resilient Back–propagation* (RProp) [9]. The evolution of the number of epochs for ANN training is a measure to avoid overfitting with the additional benefit of another critical ANN parameter being discovered by evolution.

The most interesting feature of the neuron parameter section are *Markers* – single bases (bits) which are a simple analogue to activators/repressors regulating the expression of wild–type genes. A hidden neuron marker determines, if the specific neuron associated with it is present in the decoded network. As the number of output neurons is usually strictly dependent on the specific problem the ANN should learn, these neurons are not controlled by markers.

The activation function for each neuron is composed by a maximum number of n control points of a cubic spline. In order to also allow fewer control points, each control point is "regulated" by a point marker. It should be noted that, if a neuron marker indicates the absence (or pruning) of a particular neuron, all control points for the cubic spline activation function and all connections associated with that neuron become obsolete. As a consequence, most ANN chromosomes contain noncoding regions. When using the logistic AF, only the hidden neuron markers are contained in the neuron parameter section.

ANN structure is represented by a linearized binary adjacency matrix. As in this work we are only concerned with feed–forward architectures, all elements of the upper triangle matrix must be 0, hence they are not included in the ANN genotype. Due to this linearization we are able to use the standard 2–point crossover operator for recombination. Technically, the hidden neuron markers are stored in the main diagonal of the connection matrix constructed from the genotype's structure section as shown in Fig. 3 for an exemplary network.

The maximum number of hidden neurons (neuron markers) has to be set in advance with this encoding scheme, hence, it could be labeled as *Evolutionary Pruning*, since the system imposes an upper bound on the complexity of the network.

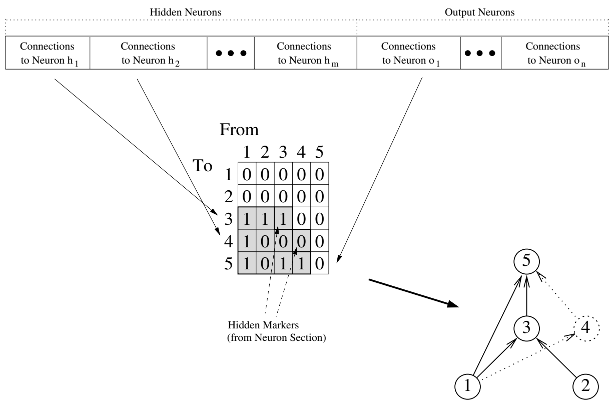

Hidden Neurons
Output Neurons
Connections to Neuron h_1
Connections to Neuron h_2
Connections to Neuron h_m
Connections to Neuron o_1
Connections to Neuron o_n
From
To
Hidden Markers (from Neuron Section)

Fig. 3. Genotype/Phenotype mapping of ANN structure.

2.1 ANN Fitness Function

The ANN fitness function comprises a complexity regularization term $\mathcal{E}_c = |C_{total}|$, with C_{total} being the set of all network connections. The *Composite Fitness Function* $\mathcal{F}$ is given by a weighted sum of *Model Fitness* and *Complexity Fitness*

$$\mathcal{F} = \alpha_1 \frac{1}{1+\mathcal{E}_m} + \alpha_2 \frac{1}{1+\mathcal{E}_c} \tag{1}$$

with $\alpha_1 + \alpha_2 = 1.0$, and $\mathcal{E}_m$ being the model error. Earlier work showed that α_2 in the range of $0.001 - 0.01$ is sufficient to guide the evolution towards ANNs of low complexity. In effect the complexity term acts as a "tie–breaker" between different ANNs with (nearly) identical model error, where the less complex network receives a slightly better fitness. In this work we set $\alpha_1 = 0.99$.

2.2 EA and ANN Parameters

The following EA and ANN parameters have been used with all the experiments in this paper:
EA Parameters: Population Size = 50, Generations = 50, Crossover Probability $p_c = 0.6$, Mutation Probability $p_m = 0.005$, Crossover = 2–Point, Selection Method = Binary Tournament.
ANN Parameters: Network Topology = Generalized Multi–Layer Perceptron, Activation Function (hidden and output neurons) = Sigmoid or evolved Cubic Splines, Output Function (all neurons) = Identity, Training = RProp, Learning Parameters Δ_0 (max 1.0), Δ_{max} (max 50.0), Number of Training Epochs (max 1000) = Evolutionary.

2.3 Cubic Spline Parameters

A cubic spline activation function is described by n control points $(x_i, y_i) \in \mathbb{R}$ where $i = 1, \ldots, n$. These n points define $n-1$ intervalls on the x-axis denoted by $x \in [x_i, x_{i+1}]$ with $x_{i+1} > x_i$. In each of these intervalls a function $f_i(x)$ is defined by

$$f_i(x) = f_i(x_i) + a_i(x - x_i) + b_i(x - x_i)^2 + c_i(x - x_i)^3 \tag{2}$$

with constants $a_i, b_i, c_i \in \mathbb{R}$. Demanding equality of the function value, and the first and second derivative at the interval borders the constants can be determined for each interval yielding a continous and differentiable function composed of a number of cubic splines.

With respect to the ANN genotype (Fig. 2) the maximum number of control points n_c has to be set prior to the evolutionary run. Also, the x–, and y–range of the cubic spline activation function has to be fixed to specific intervals $[x_{min}, x_{max}]$ (sensitivity interval) and $[a_{min}, a_{max}]$ (activation interval), respectively. However, within these bounds the evolutionary process may freely float.

For all experiments we set the number of control points $n_c = 8$ and the activation interval $[a_{min}, a_{max}] = [0.0, 1.0]$. The sensitivity interval depends on the specific problem.

3 Experimental Setup

The experiments have been devised in order to answer two main questions. The first one is simply, if the rather complex organization of the ANN genotype containing genetic information on the neurons' activation function allows the evolutionary process to find any valid and reasonable networks. Thus, like cohorts of ANN researchers, we start out with the very simple but still interesting XOR problem.

The second question to be adressed is the performance of a cubic spline network in comparison to a GMLP with conventional, sigmoidal AFs. Intuitively, the decision boundaries in classification problems could be modeled more easily by cubic splines of great plasticity than by the "frozen" logistic function. Thus, we choose the simple continous XOR problem with piece–wise linear decision boundaries, and the synthetic, but nevertheless hard *Two Spirals* problem with complex, nonlinear decision boundaries.

During the evolutionary process the ANN structure is trained on a training set and evaluated on a validation set (ANN fitness). Finally, the performance of the evolved best net is measured on a test set. Each evolutionary run is repeated 20 times.

For the XOR problem we use a network with two input and one output neuron. The binary output value 1 is defined by an activation of the output neuron $a_o > 0.5$, otherwise the output is mapped to 0. All pattern sets contain the only four example patterns.

The continous XOR (cXOR) problem is defined by a division of the unit square into four squares of identical area being separated by the lines $x = 0.5$ and $y = 0.5$. The lower left and the upper right square are labeled with binary value 1, the two remaining squares represent the class with binary value 0. The corresponding ANN has an I/O–representation identical to the simple XOR problem. The training set is composed of 100 randomly selected points in the unit square, the validation set contains 200 random patterns, and the test set is made of 10,000 equally spaced points covering the complete unit square.

The task in the *Two Spirals* problem is to discriminate between two sets of points which lie on two distinct spirals in the plane. These spirals coil three times around the origin and around one another [10]. Training, validation, and test set comprise 194 patterns each. The basic ANN structure is defined by two input neurons and two output neurons (one for each spiral, winner takes all).

For these classification problems the model error in the fitness function is simply given by $\mathcal{E}_m = \frac{e_v}{n_v}$, where e_v is the number of misclassifications on the validation set, and n_v its respective size.

4 Experimental Results

For the XOR problem we only evolved networks with cubic spline AFs. We set the maximum number of hidden neurons to three and the sensitivity interval of the neurons' cubic spline AF to $[-10.0, 10.0]$. In order to enable a more detailed fitness assignment we used $\mathcal{E}_m = MSE$ for the model error (Equ. 1). After 20 runs the average MSE of the single best nets of each run was 6.45×10^{-6}. Four of the best nets had a single hidden neuron, while the other 16 ANNs simply consisted of the two linear inputs and the output neuron with a cubic spline AF. These two simple structures are depicted in Figure 4.

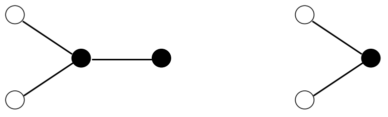


Fig. 4. Only two evolved XOR structures (white/input neuron, black/neuron with cubic spline AF).

The details of two evolved XOR nets are shown in Fig. 5.

When performing the calculations for the four possible input patterns, it can be seen that only a small portion of the sensitivity interval is used. Specifically, the big peak in each AF exceeding the activation interval seems to be useless, however, it could be argued that the steep slope provided by these peaks is profound guidance for the training algorithm.

The most interesting property of these networks is that they seem to contradict the well–known fact that a single perceptron cannot learn the XOR function

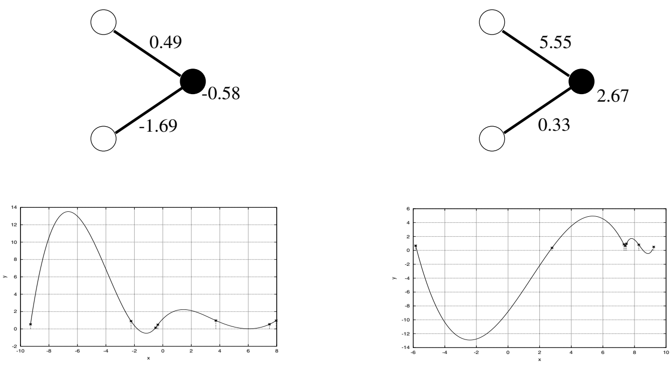


Fig. 5. Evolved XOR nets with weights, bias, and cubic spline activation function.

(Minsky's Paradox [11]). In a strict sense a perceptron is defined having a threshold AF, but even a logistic AF does not resolve Minsky's Paradox. While these functions have the desirable non–linearity, they are monotonous in contrast to the non–monotonous evolved cubic spline AFs. In all the reported aftermath of Minsky's work, it seems ironic that a simple triangle activation function would have resolved the paradox. Recently, Rueda and Oommen (2000) reported on this observation in the context of statistical pattern recognition proposing pairwise linear classifiers [12]. A non–monotonous triangle AF being a simplification of the sinusoidally shaped parts of the cubic spline AFs in Fig. 5 transforms the ANN into a pairwise linear classifier.

In Table 1 the results for the continous XOR (cXOR) problem are presented.

Table 1. *cXOR* – Structure parameters, classification accuracy (Acc) with standard deviation (StdDev) and overall best performance of evolved ANNs (averaged on 20 runs).

cXOR							
AF	Structure		Validation Set		Test Set		
	Hidden	Connections	Acc	StdDev	Acc	StdDev	Best
Sigmoid	2.80	10.70	0.9878	0.0053	0.9679	0.0065	0.9752
Spline	2.50	9.70	0.9780	0.0092	0.9583	0.0132	0.9767

Again, we set the maximum number of hidden neurons to three and the sensitivity interval of the cubic splines to $[-10.0, 10.0]$. The results slightly favor the sigmoid networks, however, the best overall ANN found is a cubic spline net (Fig. 6), even though the search space of the latter is much larger. While the genotype for the network with logistic AF consists of 39 bases, the cubic spline net chromosome comprises 538 bases.

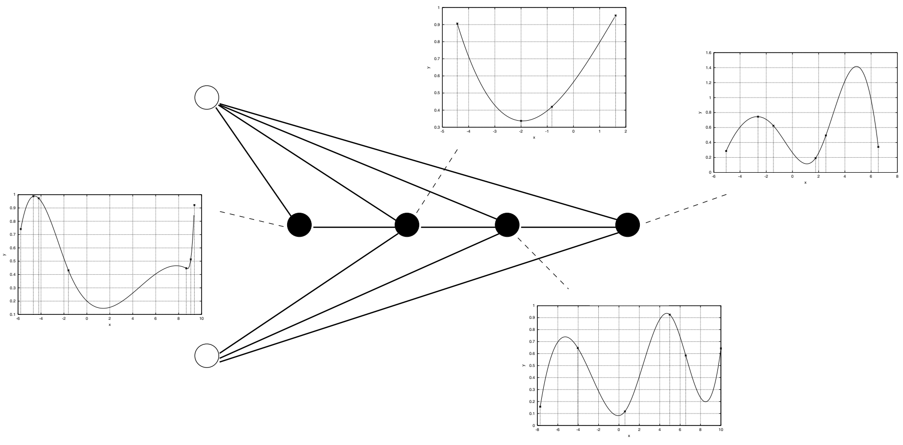


Fig. 6. *cXOR* – Overall best ANN with cubic spline AFs.

In Fig. 7 the output neuron value for the complete unit square by the best sigmoidal and the best cubic spline ANN is shown.

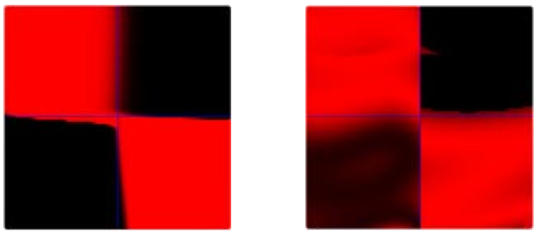


Fig. 7. *cXOR* – Output neuron activation for unit square of the best evolved sigmoid ANN (left) and the best evolved spline ANN(right)(darker = lower value).

Although, the classification results of both nets are nearly identical, it can be seen that the spline net has found a very sharp and accurate decision boundary for $x = 0.5$, and a more fuzzy boundary at $y = 0.5$. The sigmoid net exhibits uniform activations (due to monotonicity of the AF), but has slight problems to accurately model the vertical and horizontal decision boundaries.

The results of GMLPs evolved for the Two Spirals problem are presented in Table 2.

The upper bound of hidden neurons has been set to 16, and the sensitivity interval of cubic splines to $[-100.0, 100.0]$. In addition to the evolutionary runs we present results (20 different initial sets of random weights) of a standard

Table 2. *TwoSpirals* – Structure parameters, classification accuracy (Acc) with standard deviation (StdDev) and overall best performance of evolved and standard (not evolved) ANNs (averaged on 20 runs).

TwoSpirals							
AF	Structure		Validation Set		Test Set		
	Hidden	Connections	Acc	StdDev	Acc	StdDev	Best
Standard	16.00	64.00	0.6173	0.0281	0.6150	0.0303	0.6701
Sigmoid	14.90	130.70	0.8541	0.0273	0.8379	0.0327	0.8918
Spline	10.15	58.20	0.7090	0.0385	0.6791	0.0502	0.7887

sigmoid network with one hidden layer (16 neurons) for which the evolved learn parameters of the best evolved sigmoid net have been used.

Here the performance of the evolved sigmoidal nets is clearly superior to the spline networks, but the difference in search space size (234 bases sigmoid, 2682 bases spline) might be the main reason for these results. In order to gather some evidence for this explanation, we started a single run evolving spline nets for 500 generations (instead of 50) and found a network (9 hidden, 36 connections) with an accuracy of 0.8711 and 0.8969 on validation and test set, respectively. As this single run took approx. 60 hours on 22 Linux PCs (200–800MHz Pentiums), we are working on a substantial speedup of the evolutionary process by using (very) early stopping techniques for ANN training.

In Fig. 8 the classification of the unit square by the best evolved ANNs (Table 2) is depicted.

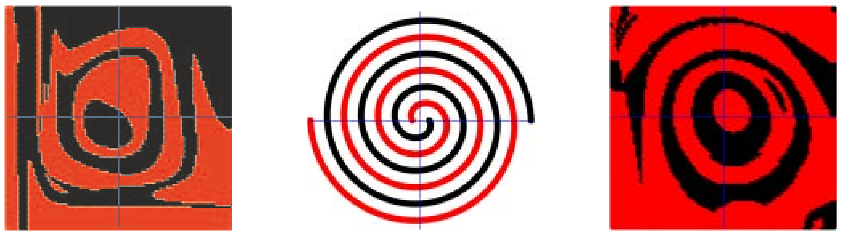


Fig. 8. *TwoSpirals* – Unit square classification by best evolved sigmoid ANN (left) and best evolved spline ANN (right), and the test set (middle).

Both evolved ANNs generate non–contigous subspaces for the two classes which make misclassifications inevitable. Though, the spline net has lower classification accuracy, the decision boundaries are of a shape closer resembling the spirals than those of the sigmoid net. However, the spirals are only defined in the areas shown in the middle figure (Fig.8), and these areas are classified more accurately by the sigmoid net. A specific source of error for the spline net is located in the upper right quadrant, where a black area is inside the inmost grey ring (falsely) covering a part of the grey spiral. As indicated above, further evolution of the complex cubic spline genes could eliminate such artefacts.

5 Summary

We have reported on experiments evolving structure and activation functions of generalized multi–layer perceptrons. We compared evolution of ANNs with the widely used logistic activation function in each hidden and output neuron to networks with cubic spline activation functions. Each neuron in the spline net can have a different nonlinear, non–monotonous activation function of (nearly) arbitrary shape. The non–monotonicity of these activation functions led the way to solve the XOR problem with only a single neuron which is impossible by using monotonous activation functions.

Further experiments with the more complex two spirals problem favored the evolved sigmoid nets. However, the one order of magnitude larger genotype of the spline networks could be the main reason for this result. By inspecting the decision boundaries of the two types of networks the potential of the spline nets to model linear and circular boundaries of class subspaces with great accuracy have been demonstrated.

Also, with the two spirals problem the number of hidden neurons and connections of a spline network was found to be considerably smaller than that of a sigmoid network. This is in accordance to reports on the adaptive spline neural networks presented in [5].

Continuing work will explore the use of cubic splines with improved locality, as a single mutation in a cubic spline gene can dramatically change the shape of the whole activation function. More advanced classes of cubic splines avoid this behavior, and a change of a single control point will result in only a local change of the cubic spline in the neighborhood of this point. Moreover, techniques to decrease the training time of ANNs are currently investigated so as to decrease the running time of network evolution employing the parallel netGEN system.

References

1. Lippmann, R.P.: An introduction to computing with neural nets. IEEE Acoustics, Speech and Signal Processing **4** (1987) 4–22
2. Cybenko, G.: Approximation by superpositions of a sigmoidal function. Mathematics of Control, Signals, and Systems **2** (1987) 303–314
3. Barron, A.R.: Universal approximation bounds for superpositions of a sigmoidal function. IEEE Transactions on Information Theory **39** (1993) 930–944
4. Liu, Y., Yao, X.: Evolutionary Design of Artificial Neural Networks with Different Nodes. In: Proceedings of the Third IEEE International Conference on Evolutionary Computation. (1996) 570–675
5. Vecci, L., Piazza, F., Uncini, A.: Learning and approximation capabilities of adaptive spline activation function neural networks. Neural Networks (1998) 259–270

6. Sopena, J.M., Romero, E., Alquézar, R.: Neural networks with periodic and monotonic activation functions: a comparative study in classification problems. In: Proceedings of the 9th International Conference on Artificial Neural Networks. (1999)
7. Zell, A., Mamier, G., Vogt, M., Mach, N., Huebner, R., Herrmann, K.U., Soyez, T., Schmalzl, M., Sommer, T., Hatzigeogiou, A., Doering, S., Posselt, D.: SNNS Stuttgart Neural Network Simulator, User Manual. University of Stuttgart (1994)
8. Miller, G.F., Todd, P.M., Hegde, S.U.: Designing neural networks using genetic algorithms. In Schaffer, J.D., ed.: Proceedings of the Third International Conference on Genetic Algorithms, San Mateo, California, Philips Laboratories, Morgan Kaufman Publishers, Inc. (1989) 379–384
9. Riedmiller, M., Braun, H.: A direct adaptive method for faster backpropagation learning: The RPROP algorithm. In: Proceedings of the IEEE International Conference on Neural Networks, San Francisco, CA (1993)
10. CMU-Repository: ftp://ftp.cs.cmu.edu/afs/cs.cmu.edu/ project/connect/bench/. WWW Repository, Carnegie Mellon University (1993)
11. Minsky, M.L., Papert, S.A.: Perceptrons: Introduction to Computational Geometry. Expanded edn. MIT Press (1988)
12. Rueda, L., Oommen, B.J.: The Foundational Theory of Optimal Bayesian Pairwise Linear Classifiers. In: Proceedings of Joint IAPR International Workshops SSPR 2000 and SPR 2000 (LNCS 1876), Springer (2000) 581–590

Multilingual Document Clustering, Topic Extraction and Data Transformations

Joaquim Silva[1], João Mexia[2], Carlos A. Coelho[3], and Gabriel Lopes[1]

[1] DI / FCT / Universidade Nova de Lisboa
Quinta da Torre, 2725 Monte da Caparica, Portugal
{jfs, gpl}@di.fct.unl.pt
[2] DM / FCT / Universidade Nova de Lisboa
Quinta da Torre, 2725 Monte da Caparica, Portugal
[3] DM / ISA / Universidade Técnica de Lisboa,
Tapada da Ajuda, Portugal
coelho@madpet.isa.pt

Abstract. This paper describes a statistics-based approach for clustering documents and for extracting cluster topics. Relevant Expressions (REs) are extracted from corpora and used as clustering base features. These features are transformed and then by using an approach based on Principal Components Analysis, a small set of document classification features is obtained. The best number of clusters is found by Model-Based Clustering Analysis. Data transformations to approximate to normal distribution are done and results are discussed. The most important REs are extracted from each cluster and taken as cluster topics.

1 Introduction

We aimed at developing an approach for automatically separating documents from multilingual corpora into clusters. We required no prior knowledge about document subject matters or language and no morphosyntactic information was used, since we wanted a language independent system. Moreover, we also wanted to extract the main topics characterizing the documents in each cluster.

Clustering software usually needs a matrix of objects characterized by a set of features. In order to obtain those features, we used LocalMaxs algorithm to automatically extract REs from corpora [1]. These REs, such as *Common agriculture policy*, *Common Customs*, etc., provide important information about the content of the texts. Therefore we decided to use them as *base features* for clustering. The most informative REs correspond to cluster topics, as we will show. This paper is organized as follows: features extraction is explained in section 2; data transformations to approximate to normality, clustering and summarization in sections 3 and 4; section 5 presents and discusses results obtained; related work and conclusions are drawn in section 6 and 7.

P. Brazdil and A. Jorge (Eds.): EPIA 2001, LNAI 2258, pp. 74–87, 2001.

2 Extracting Multiword Features from Corpora

We used LocalMaxs algorithm, Symmetric Conditional Probability (SCP) statistical measure and Fair Dispersion Point Normalization (FDPN) to extract REs from any corpus. A full explanation of these tools is given in [1]. However, a brief description is presented here. Thus, let us consider that an n-gram is a string of words in any text[1]. For example the word *rational* is an 1-gram; the string *rational use of energy* is a 4-gram. LocalMaxs is based on the idea that each n-gram has a kind of cohesion sticking the words together within the n-gram. Different n-grams usually have different cohesion values. One can intuitively accept that there is a strong cohesion within the n-gram (*Bill, Clinton*) i.e. between the words *Bill* and *Clinton*. However, one cannot say that there is a strong cohesion within the n-gram (*or, if*) or within the (*of, two*). Thus, the $SCP(.)$ cohesion value of a generic bigram (x, y) is obtained by

$$SCP((x,y)) = p(x|y).p(y|x) = \frac{p(x,y)^2}{p(x).p(y)} \tag{1}$$

where $p(x, y)$, $p(x)$ and $p(y)$ are the probabilities of occurrence of bigram (x, y) and unigrams x and y in the corpus; $p(x|y)$ stands for the conditional probability of occurrence of x in the first (left) position of a bigram, given that y appears in the second (right) position of the same bigram. Similarly $p(y|x)$ stands for the probability of occurrence of y in the second (right) position of a bigram, given that x appears in the first (left) position of the same bigram.

However, in order to measure the cohesion value of each n-gram of any size in the corpus, FDPN concept is applied to $SCP(.)$ measure and a new cohesion measure, $SCP\_f(.)$, is obtained.

$$SCP\_f((w_1 \dots w_n)) = \frac{p((w_1 \dots w_n))^2}{\frac{1}{n-1}\sum_{i=1}^{n-1} p(w_1 \dots w_i).p(w_{i+1} \dots w_n)} \tag{2}$$

where $p((w_1 \dots w_n))$ is the probability of the n-gram $w_1 \dots w_n$ in the corpus. So, any n-gram of any length is "transformed" in a pseudo-bigram[2] that reflects the *average cohesion* between each two adjacent contiguous sub-n-gram of the original n-gram. Then, LocalMaxs algorithm elects every n-gram whose $SCP\_f(.)$[3] cohesion value is a salient one (a local maximum), (see [1] for details). See some examples of elected n-grams, i.e., REs: *Human Rights*, *Human Rights in East Timor*, *politique énergétique* and *economia energética*.

[1] We use the notation $(w_1 \dots w_n)$ or $w_1 \dots w_n$ to refer an n-gram of length n.

[2] Roughly we can say that known statistical cohesion / association measures such as $Dice(.)$, $MI(.)$, χ^2, etc. seem to be "tailored" to measure just 2-grams. However, by applying FDPN to those measures, it is possible to use them for measuring the cohesion values of n-grams for any value of n [1].

[3] LocalMaxs has been used in other applications with other statistical measures, as it is shown in [1]. However, for Information Retrieval (IR) purposes, very interesting results were obtained by using $SCP\_f(.)$, in comparison with other measures [2].

2.1 The Number of Features

Document clustering usually requires a matrix of documents characterized by the smallest possible set of variables. That matrix is then conveyed to clustering software. So, a multilingual 872,795 words corpus with 324 documents[4] was used in order to test our approach. LocalMaxs extracted 25,838 REs from that corpus. Obviously, we cannot use such a high number of features for distinguishing such a small number of objects (324 documents). However, these REs (base features) provide the basis for building a new and reduced set of features.

2.2 Reducing the Number of Features

Let us take the following extracted REs containing the word *agricultural*: *agricultural products*, *processing of agricultural products*, *agricultural products receiving refunds* and *common agricultural policy*. For document clustering purposes, there is redundancy in these REs, since, for example, whenever *processing of agricultural products* is in a document, *agricultural products* is also in the same document and it may happen that, *common agricultural policy* is also in that document. These redundancies can be used to reduce the number of features.

Thus, according to Principal Components Analysis (PCA), often the original m correlated random variables (features) $X_1, X_2, \ldots, X_m$ can be "replaced" by a subset $Y_1, Y_2, \ldots, Y_k$ of the m new *uncorrelated* variables (components) $Y_1, Y_2, \ldots, Y_m$, each one being a linear combination of the m original variables, i.e., those k *principal components* provide most of the information of the original m variables [5, pages 340-350]. The original data set, consisting of l measurements of m variables, is reduced to another one consisting of l measurements of k principal components. Principal components depend solely on the covariance matrix $\boldsymbol{\Sigma}$ (or the correlation matrix $\boldsymbol{\rho}$) of the original variables $X_1, \ldots, X_m$. Now we state $RE_1, RE_2, \ldots, RE_p$ as being the original p variables (REs) of the sub-ELIF corpus. Then, for a reduced set of new variables (principal components) we would have to estimate the associated covariance matrix of the variables $RE_1, \ldots, RE_p$. So, let the sample covariance matrix $\boldsymbol{RE}$ be the estimator of $\boldsymbol{\Sigma}$.

$$\boldsymbol{RE} = \begin{bmatrix} RE_{1,1} & RE_{1,2} & \ldots & RE_{1,p} \\ RE_{1,2} & RE_{2,2} & \ldots & RE_{2,p} \\ \vdots & \vdots & \ddots & \vdots \\ RE_{1,p} & RE_{2,p} & \ldots & RE_{p,p} \end{bmatrix} \tag{3}$$

where $RE_{i,k}$ estimates the covariance $Cov(RE_i, RE_k)$. $\boldsymbol{RE}$ can be seen as a similarity matrix between REs. Unfortunately, due to this matrix huge size $(25,838 \times 25,838)$, we cannot obtain principal components using available software. Moreover it is unlikely that PCA could achieve the reduction we need: from $25,838$ original features to $k < 324$ (the number of documents) new features (principal components).

[4] This is part of the European Legislation in Force (sub-ELIF) corpus: http://europa.eu.int/eur-lex.

2.3 Geometrical Representations of Document Similarity

We can associate to the jth document, the vector $\boldsymbol{d}_j^T = [x_{1,j}, \ldots, x_{p,j}]$[5] where $x_{i,j}$ is the original number of occurrences of the ith RE in the jth document[6]. Now, we can have a smaller (324×324) covariance matrix

$$\boldsymbol{S} = \begin{bmatrix} S_{1,1} & S_{1,2} & \ldots & S_{1,n} \\ S_{1,2} & S_{2,2} & \ldots & S_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ S_{1,n} & S_{2,n} & \ldots & S_{n,n} \end{bmatrix} \tag{4}$$

where

$$S_{j,l} = \frac{1}{p-1} \sum_{i=1}^{i=p} (x_{i,j} - x_{\cdot,j})(x_{i,l} - x_{\cdot,l}) \tag{5}$$

and $x_{\cdot,j}$, meaning the average number of occurrences per RE in the jth document, is given by[7]

$$x_{\cdot,j} = \frac{1}{p} \sum_{i=1}^{i=p} x_{i,j} \ . \tag{6}$$

Then $\boldsymbol{S}$ will be a similarity matrix between documents.

Escoufier and L'Hermier [3] proposed an approach, based on PCA, to derive geometrical representations from similarity matrices. Since $\boldsymbol{S}$ is symmetric we have $\boldsymbol{S} = \boldsymbol{P\Lambda P}^T$, with $\boldsymbol{P}$ orthogonal ($\boldsymbol{P} = [\boldsymbol{e}_1, \ldots, \boldsymbol{e}_n]$, the matrix of normalized eigenvectors of $\boldsymbol{S}$) and $\boldsymbol{\Lambda}$ diagonal. The principal elements of $\boldsymbol{\Lambda}$ are the eigenvalues $\lambda_1, \ldots, \lambda_n$ of $\boldsymbol{S}$ and $\lambda_1 \geq \lambda_2 \cdots \geq \lambda_n \geq 0$. Thus $\boldsymbol{S} = \boldsymbol{QQ}^T$ with

$$\boldsymbol{Q} = \boldsymbol{P\Lambda}^{1/2} \ . \tag{7}$$

The elements of the ith line of $\boldsymbol{Q}$ will be the coordinates of the point associated with the ith document. We may consider only the coordinates corresponding to the leading eigenvalues. Then, to assess how much of the total information is carried out by the first k components, i.e. the first k columns of $\boldsymbol{Q}$, we may use

$$PTV(k)^8 = \frac{\sum_{j=1}^{j=k} \lambda_j}{\sum_{j=1}^{j=n} \lambda_j} \ . \tag{8}$$

So, by taking the first k columns of matrix $\boldsymbol{Q}$ such that $PTV(k)$ equals, say .80 or more, we can reduce the initial large number of features to $k < n$ new features (components). However, considering the 324 documents of the corpus, if we use the original number of occurrences of the ith RE in the jth document ($x_{i,j}$) to obtain "similarities" (see (5)), we need the first 123 components to provide

[5] We will use p for the number of REs of the corpus and n for the number of documents.
[6] These numbers of occurrences can be transformed, as we will see in Sect. 2.4.
[7] When we replace an index by a dot, a mean value has been obtained.
[8] PTV are initials for cumulative Proportion of the Total Variance.

0.85 of the total information, i.e. $PTV(123) = 0.85$. Although it corresponds to 0.4% of the initial 25, 838 features, large numbers of components must be avoided in order to minimize computational effort of the clustering process. So, to reduce this number, we need to "stimulate" similarities between documents, and therefore, the original occurrences of the REs in documents must be transformed.

2.4 Transforming the Original Number of Occurrences

As referred above, the geometrical representation may be obtained from transformed occurrences. The technique we used has four phases. In the first phase we standardize in order to correct document heterogeneity. This heterogeneity is measured by the variation between the occurrence numbers of the different REs inside each document. This variation may be assessed by

$$V(D_j) = \frac{1}{p-1}\sum_{i=1}^{i=p}(x_{i,j} - x_{\cdot,j})^2 \qquad j = 1, \ldots, n \tag{9}$$

where $x_{\cdot,j}$ is given by (6). The standardized values will be

$$z_{i,j} = (x_{i,j} - x_{\cdot,j}).[V(D_j)]^{-1/2} \qquad i = 1, \ldots, p;\ \ j = 1, \ldots, n\ . \tag{10}$$

In the second phase we evaluate the variation between documents for each RE. Although, each RE associated variation must reflect how much the RE occurrences vary in different documents, due to document content, not due to document size. Therefore we use normalized values to calculate this variation:

$$V(RE_i) = \frac{1}{n-1}\sum_{j=1}^{j=n}(z_{i,j} - z_{i,\cdot})^2 \qquad i = 1, \ldots, p\ . \tag{11}$$

These values are important since we found that, generally, the higher the $V(RE_i)$, the more information is carried out by the ith RE. On the other hand, it was observed that REs constituted by long words, usually are more informative from the IR / Text Mining point of view (e.g. *agricultural products* or *communauté economique européenne* are more informative than *same way*, *plus au moins* or *reach the level*). Thus, in a third phase we define *weighted occurrences* as

$$x^*_{i,j} = x_{i,j} \cdot V(RE_i) \cdot AL(RE_i) \qquad i = 1, \ldots, p;\ \ j = 1, \ldots, n \tag{12}$$

where $AL(RE_i)$ is the average number of characters per word in the ith RE. Lastly, in the fourth phase we carry out a second standardization considering the *weighted occurrences*. This is for correcting document size heterogeneity, since we do not want that the document size affects its relative importance. Thus

$$z^*_{i,j} = (x^*_{i,j} - x^*_{\cdot,j}).[V(D^*_j)]^{-1/2} \qquad i = 1, \ldots, p;\ \ j = 1, \ldots, n \tag{13}$$

$$\text{where} \qquad V(D^*_j) = \frac{1}{p-1}\sum_{i=1}^{i=p}(x^*_{i,j} - x^*_{\cdot,j})^2 \qquad j = 1, \ldots, n\ . \tag{14}$$

These standardizations are *transformed occurrences* and are used to obtain the similarity matrix between documents, whose generic element is given by

$$S_{j,l} = \frac{1}{p-1}\sum_{i=1}^{i=p}(z^*_{i,j} - z^*_{\cdot,j})(z^*_{i,l} - z^*_{\cdot,l}) \qquad j = 1,\ldots,n\,;\;\; l = 1,\ldots,n\ . \tag{15}$$

2.5 Non-informative REs

Some high-frequency REs appearing in most documents written in the same language are not informative from the Text Mining point of view, e.g., locutions such as *Considérant que* (*having regard*), *and in particular*, or other expressions which are incorrect REs, such as *of the* or *dans les* (*in the*). Although these expressions are useless to identify document topics, they are informative for distinguishing different languages. As a matter of fact they occur in most documents of the same language, and their associated variation (see (11)) is usually high or very high, i.e., they are relevant to "approximate" documents of the same language for calculating similarities between documents (see (12), (13) and (15)).

So, it seems that either they should be removed to distinguish topics in documents of the same language, or they should be kept for distinguishing documents of different languages. To solve this problem, we use the following criterion: the REs having at least one extremity (the leftmost or the rightmost word) that exists in at least 90 % of the documents we are working with, are removed from the initial set of REs. We follow that criterion since these expressions usually begin or end with words occurring in most documents of the same language, e.g., *of*, *les*, *que*, etc.. As we will see in Subsect. 3.3, the documents and REs with which the system is working, depend on the node of the clustering tree.

To summarize, in this section we obtained a matrix where a small set of components characterizes a group of documents. This matrix will be used as input for clustering. For this purpose, the matrix of document similarity ($\boldsymbol{S}$) (see (4)) was calculated. Its generic element is given by (15). Then, from $\boldsymbol{S}$, $\boldsymbol{Q}$ was obtained by (7) and the first k columns of $\boldsymbol{Q}$ were taken, such that $PTV(k) \geq 0.80$. Finally, the latter matrix will be conveyed to clustering software.

Considering the initial $25,838$ REs for the 324 documents of the sub-ELIF corpus, we obtained $PTV(3) = .848$; $PTV(5) = .932$ and $PTV(8) = .955$.

3 Clustering Documents

We need to split documents into clusters. However we do not know how many clusters should be obtained. Moreover, though we have obtained k features (components) to evaluate the documents, we do not know neither the composition of each cluster, nor its volume, shape and orientation in the k-axes space.

3.1 The Model-Based Cluster Analysis

Considering the problem of determining the structure of clustered data, without prior knowledge of the number of clusters or any other information about their composition, Fraley and Raftery [4] developed the Model-Based Clustering Analysis (MBCA). By this approach, data are represented by a mixture model where each element corresponds to a different cluster. Models with varying geometric properties are obtained through different Gaussian parameterizations and cross-cluster constraints. This clustering methodology is based on multivariate normal (Gaussian) mixtures. So the density function associated to cluster c has the form

$$f_c(\boldsymbol{x}_i|\boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c) = \frac{exp\{-\frac{1}{2}(\boldsymbol{x}_i - \boldsymbol{\mu}_c)^T \boldsymbol{\Sigma}_c^{-1}(\boldsymbol{x}_i - \boldsymbol{\mu}_c)\}}{(2\pi)^{\frac{r}{2}} \, |\boldsymbol{\Sigma}_c|^{\frac{1}{2}}} \quad . \tag{16}$$

Clusters are ellipsoidal, centered at the means $\boldsymbol{\mu}_c$; element $\boldsymbol{x}_i$ belongs to cluster c. The covariance matrix $\boldsymbol{\Sigma}_c$ determines other geometric characteristics. This methodology is based on the parameterization of the covariance matrix in terms of eigenvalue decomposition in the form $\boldsymbol{\Sigma}_c = \lambda_c \boldsymbol{D}_c \boldsymbol{A}_c \boldsymbol{D}_c^T$, where $\boldsymbol{D}_c$ is the matrix of eigenvectors, determining the orientation of the principal components of $\boldsymbol{\Sigma}_c$. $\boldsymbol{A}_c$ is the diagonal matrix whose elements are proportional to the eigenvalues of $\boldsymbol{\Sigma}_c$, determining the shape of the ellipsoid. The volume of the ellipsoid is specified by the scalar λ_c. Characteristics (orientation, shape and volume) of distributions are estimated from the input data, and can be allowed to vary between clusters, or constrained to be the same for all clusters. Considering our application, input data is given by the k columns of matrix $\boldsymbol{Q}$ (see (7) and (8)).

MBCA subsumes the approach with $\boldsymbol{\Sigma}_c = \lambda \boldsymbol{I}$, long known as *k-means*, where sum of squares criterion is used, based on the assumption that all clusters are spherical and have the same volume (see Table 1). However, in the case of k-means, the number of clusters has to be specified in advance, and considering many applications, real clusters are far from spherical in shape. Therefore we have chosen MBCA for clustering documents. Then, function `emclust` has been used with `S-PLUS` package, which is available for `Windows` and `Linux`.

During the cluster analysis, `emclust` shows the Bayesian Information Criterion (BIC), a measure of evidence of clustering, for each "pair" *model-number of clusters*. These "pairs" are compared using BIC: the larger the BIC, the stronger the evidence of clustering (see [4]). The problem of determining the number of clusters is solved by choosing the *best model*. Table 1 shows the different models

Table 1. Parameterizations of the covariance matrix $\boldsymbol{\Sigma}_c$ in the Gaussian model and their geometric interpretation

$\boldsymbol{\Sigma}_c$	Distribution	Volume	Shape	Orientation
$\lambda \boldsymbol{I}$	Spherical	Equal	Equal	
$\lambda_c \boldsymbol{I}$	Spherical	Variable	Equal	
$\lambda \boldsymbol{D} \boldsymbol{A} \boldsymbol{D}^T$	Ellipsoidal	Equal	Equal	Equal
$\lambda_c \boldsymbol{D}_c \boldsymbol{A}_c \boldsymbol{D}_c^T$	Ellipsoidal	Variable	Variable	Variable
$\lambda \boldsymbol{D}_c \boldsymbol{A} \boldsymbol{D}_c^T$	Ellipsoidal	Equal	Equal	Variable
$\lambda \boldsymbol{D}_c \boldsymbol{A} \boldsymbol{D}_c^T$	Ellipsoidal	Variable	Equal	Variable

used during the calculation of the *best model*. Models must be specified as a parameter of the function `emclust`. However, usually there is no prior knowledge about the model to choose. Then, by specifying all models, `emclust` gives us BIC values for each pair *model-number of clusters* and proposes the *best model* which indicates which cluster must be assigned to each object (document).

3.2 Assessing Normality of Data. Data Transformations

MBCA works based on the assumption of normality of data. Then, Gaussian distribution must be checked for the univariate marginal distributions of the documents on each component. Thus, a QQ-plot is made for each component. For this purpose, each of the first k columns of matrix $\boldsymbol{Q}$ (see (7) and (8)) is standardized, ordered and put on y axis of the QQ-plot. Then, standardized normal quantiles are generated and put on x axis of the QQ-plot (see [5, pages 146-162] for details). Fig. 1(a) represents the QQ-plot for the 2th component, assessing the normality of data of cluster 1.2[9]. This QQ-plot is representative, since most of the components for other clusters produced similar QQ-plots. Most

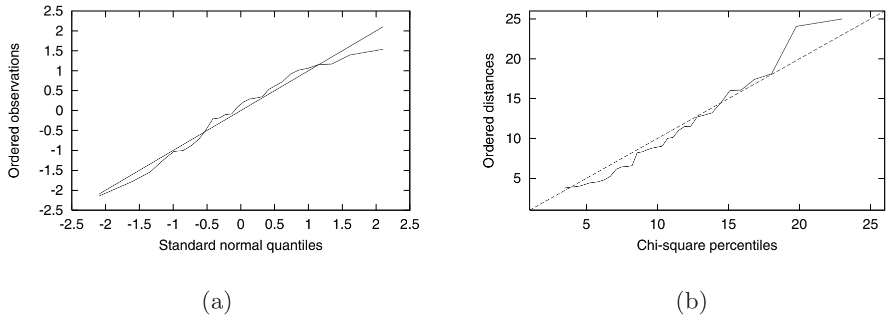


Fig. 1. QQ-plot of data for cluster 1.2 on 2nd component (a); Chi-square plot of the ordered distances for data in cluster 1.2 (b)

of the times, if QQ-plots are straight (univariate distributions are normal), the joint distribution of the k dimensions (components) are multivariate normal. However, multivariate normality must be tested. Then, a Chi-square plot is constructed for each cluster. Thus, the square distances are ordered from smallest to largest as $d^2_{(1)} \leq d^2_{(2)} \leq \cdots \leq d^2_{(m)}$, where $d^2_{(j)} = (\boldsymbol{x}_j - \overline{\boldsymbol{x}}_c)^T \boldsymbol{S}_c^{-1} (\boldsymbol{x}_j - \overline{\boldsymbol{x}}_c)$. V ector $\boldsymbol{x}_j$ is the jth element (document) of cluster c; $\overline{\boldsymbol{x}}_c$ is the means vector for the k dimensions of that cluster, and $\boldsymbol{S}_c^{-1}$ is the inverse of the estimator of the cluster covariance matrix. Then the pairs $(d^2_{(j)}, \chi^2_k((j-1/2)/m))$ are graphed,

[9] In Sect. 5 we will deal with specific clusters.

where m is the number of elements of the cluster and $\chi^2_k((j-1/2)/m)$ is the $100(j-1/2)/m$ percentile of the Chi-square distribution with k (the number of components, see (8)) deg rees of freedom. The plot should resemble a straight line. A systematic curved pattern suggests lack of normality. Since the straightness of the univariate and multivariate plots are not perfect (see Fig. 1(a) and Fig. 1(b)), some transformations of the data were tried to approximate to normality. Transformations were applied to some of the k columns of matrix $\boldsymbol{Q}$ (see (7) and (8)): those whose QQ-plot suggested lack of normality. So, let y be an arbitrary element of a given column we want to transform. We considered the following slightly modified family of power transformations from y to $y^{(\lambda)}$ [6]:

$$y^{(\lambda)} = \begin{cases} \frac{y^{\lambda}-1}{\lambda} & \lambda \neq 0 \\ \ln y & \lambda = 0 \end{cases} . \tag{17}$$

Power transformations are defined only for positive variables. However this is not restrictive, because a single constant can be added to each element in the column if some of the elements are negative. Thus, given the elements $y_1, y_2, \ldots, y_m$ in a given column, the Box-Cox [6] solution for the choice of an appropriate power λ for that column is the one which maximizes the expression

$$l(\lambda) = -\frac{m}{2} \ln \left[\frac{1}{m} \sum_{j=1}^{j=m} (y_j^{(\lambda)} - \overline{y^{(\lambda)}})^2 \right] + (\lambda - 1) \sum_{j=1}^{j=m} \ln y_j \tag{18}$$

$$\text{where} \qquad \overline{y^{(\lambda)}} = \frac{1}{m} \sum_{j=1}^{j=m} y_j^{(\lambda)} \tag{19}$$

and m is the number of elements of the cluster; $y_j^{(\lambda)}$ is defined in (17). So, every element y of the ith column of matrix $\boldsymbol{Q}$ will be transformed from y to $y^{(\lambda)}$ according to (17) where $\lambda = \hat{\lambda}_i$, i.e., the value of λ maximizing $l(\lambda)$. This new matrix was conveyed to clustering software. However, the results obtained were not better than using the non-transformed matrix, as we will see in Sect. 5. Therefor e, although the straightness of the plots in Fig. 1(a) and Fig. 1(b) are not perfect, as it should be to ensure normality, it is close enough to encourage us to use MBCA as an adequate approach for clustering this kind of data.

3.3 Sub-clusters

As we will see in Sect. 5, our approach organizes sub-ELIF corpus in 3 main clusters: English, Portuguese and French documents. However, we can distinguish different subjects in different documents of the same cluster. So, a hierarchical tree of clusters is built as follows: let us consider that every cluster in the tree is a node. For every node, non-informative REs are removed (see Subsect. 2.5) from the set of REs contained in the documents of that node (a subset of the original REs), in order to obtain a new similarity matrix between documents (see (15)). Then, the first k columns of the new matrix $\boldsymbol{Q}$ are taken (see (7) and (8)), and new clusters are proposed by MBCA.

3.4 Choosing the Best Number of Clusters

As has been said, MBCA calculates the *best model* based on the k first columns of matrix $\boldsymbol{Q}$ (k components). A large number of components means no significant information loss, which is important for a correct clustering (*best model*) to be proposed by MBCA. On the other hand, a large number of components must be avoided, since it takes MBCA to estimate large covariance matrices – during the internal computation for different models – which can be judged to be close to singularity (see [4]). Therefore, we use the following criterion: the first k components are chosen in such a way that $PTV(k) \geq 0.80$ (see (8)).

Then, based on those k components, MBCA produces a list of BIC values for each model: *VVV (Variable volume, Variable shape, Variable Orientation), EEV* etc. (see Table 1). Each list may have several local maxima. The largest local maximum over all models is usually proposed as the *best model*. However, a heuristic that works well in practice (see [4]) chooses the number of clusters corresponding to the first decisive local maximum over all the models considered.

4 Summarization

Summarizing a document and summarizing a cluster of documents are different tasks. As a matter of fact, documents of the same cluster have common REs such as *rational use of energy* or *energy consumption*, rather than long sentences which are likely to occur in just one or two documents. Then, summarizing topics seems adequate to disclose the core content of each cluster.

Cluster topics correspond to the most important REs in the cluster. Let the cluster from where we want to extract its topics be the “target cluster”. Then, in order to extract it, first the REs of the *parent node* of the target cluster are ordered according to the value given by $Score(RE_i)$ assigned to the ith RE.

$$Score(RE_i) = Thr(RE_i) \cdot V(RE_i) \cdot AL(RE_i) \qquad \text{where} \tag{20}$$

$$Thr(RE_i) = \begin{cases} 1 & SCP\_f(RE_i) \geq threshold \\ 0 & \text{else} \end{cases} . \tag{21}$$

$V(RE_i)$ and $AL(RE_i)$ have the same meaning as in (12). Thus, $Thr(\cdot)$ corresponds to a filter that “eliminates” REs whose $SCP\_f(\cdot)$ cohesion value (see Sect. 2) is lower than *threshold* – a value empirically set to 0.015. These REs, e.g., *Considérant que*, *and in particular*, etc., are not informative for IR / Text Mining. Most of the times, these REs are previously eliminated when selecting the informative REs for calculating the covariance matrix; however it may not happen in case of a multilingual set of documents. (see Subsect. 2.5).

So, the largest $Score(RE_i)$ corresponds to the most important RE. For example, according with this criterion, the 15 most important REs of the initial cluster (the one containing all documents) are the following: *Nomenclatura Combinada (Combined Nomenclature)*, *nomenclature combinée*, *Member States*, *combined nomenclature*, *États membres (Member states)*, *Council Regulation*, *produtos agrícolas (agricultural products)*, *produits agricoles*, *utilização racional (rational use)*, *nomenclature tarifaire (tariff nomenclature)*, *autoridades aduan eiras (customs authorities)*, *estância aduaneira (customs office)*, *Common Customs*, *indicados na coluna (indicated in column)* and *agricultural products.*

Now, taking for instance the "French documents" as the target cluster, we cannot "guarantee" that *produits agricoles* will be a topic, since not every French document content is about *produits agricoles.* On the other hand, the same topic often appears in different documents, written in different forms, (e.g. *produits agricoles* and *Produits Agricoles*). Hence, according to $Score(\cdot)$, the 15 most important REs of the target cluster occurring in at least 50 % of its documents are put in a list. From this list, the REs with $Score(\cdot)$ value not lower than 1/50 of the maximum $Score(\cdot)$ value obtained from that list, are considered topics.

5 Results

Sub-ELIF is a multilingual *corpus* with 108 documents per language. For each document there are two other documents which are translations to the other languages. From Table 2 we can see the hierarchical tree of clusters obtained by this approach. We also present the topics extracted from each cluster.

Table 2. Evaluation of the clusters

Cluster	Main topic	Correct #	Total #	Act. corr. #	Prec. (%)	Rec. (%)
1	european communities	108	108	108	100	100
2	Comunidade Europeia	108	107	107	100	99.1
3	Communauté européenne	108	109	108	99.1	100
1.1	rational use of energy	23	23	20	86.9	86.9
1.2	agricultural products	27	27	21	77.8	77.8
1.3	combined nomenclature	58	58	51	87.9	87.9
2.1	economia de energia	23	26	21	80.8	91.3
2.2	produtos agrícolas	27	25	21	84	77.8
2.3	Nomenclatura Combinada	58	56	52	92.9	89.7
3.1	politique énergétique	23	26	22	84.6	95.7
3.2	produits agricoles	27	27	21	77.8	77.8
3.3	nomenclature combinée	58	56	53	94.6	91.4

Cluster 1: *European Communities*, *Member States*, *EUROPEAN COMMUNITIES*, *Council Regulation*, *Having regard to Council Regulation*, *Having regard to Council* and *Official Journal.*

Cluster 2: *Comunidade Europeia*, *Nomenclatura Combinada*, *COMUNIDADES EUROPEIAS* and *directamente aplicável (directly applicable).*

Cluster 3: *Communauté européenne*, *nomenclature combinée*, *États membres*, *COMMUNAUTÉS EUROPÉENNES* and *directement applicable.*

Cluster 1.1: *rational use of energy*, *energy consumption* and *rational use.*

Cluster 1.2: *agricultural products*, *Official Journal*, *detailed rules*, *Official Journal of the European Communities*, *proposal from the Commission*, *publication in the Official Journal* and *entirely and directly.*

Cluster 1.3: *combined nomenclature*, *Common Customs*, *customs authorities*, *No 2658/87*, *goods described*, *general rules*, *appropriate CN*, *Having regard to Council Regulation*, *tariff and statistical* and *Customs Code.*

Cluster 2.1: *economia de energia (energy saving)*, *utilização racional*, *racional da energia (rational of energy)* and *consumo de energia (energy consuming).*

Cluster 2.2: *produtos agrícolas*, *Comunidades Europeias*, *Jornal Oficial*, *directamente aplicável*, *COMUNIDADES EUROPEIAS*, *Jornal Oficial das Comunidades*, *directamente aplicável em todos os Estados-membros (directly applicable to all Member States)*, *publicação no Jornal Oficial*, *publicação no Jornal Oficial das Comunidades* and *Parlamento Europeu (European Parliament).*

Cluster 2.3: *Nomenclatura Combinada*, *autoridades aduaneiras (customs authorities)*, *indicados na coluna*, *mercadorias descritas*, *informações pautais vinculativas (binding tariff informations)*, *Aduaneira Comum (Common Customs)*, *regras gerais*, *códigos NC (NC codes)* and *COMUNIDADES EUROPEIAS.*

Cluster 3.1: *politique énergétique (energy policy)*, *rationnelle de énergie* and *l'utilization rationnelle.*

Cluster 3.2: *produits agricoles*, *organisation commune*, *organisation commune des marchés*, *directment applicable*, *Journal officiel*, *Journal officiel des Communautés* and *COMMUNAUTÉS EUROPÉENNES.*

Cluster 3.3: *nomenclature combinée*, *autorités douaniès (customs authorities)*, *nomenclature tarifaire*, *No 2658/87*, *marchandises décrites*, *tarifaire et statistique (tariff and statistical)* and *COMMUNAUTÉS EUROPÉENNES.*

5.1 Discussion

MBCA (function `emclust`) proposed clusters 1, 2 and 3 considering EEV as the *best model*, (see Table 1). EEV model was also considered *best model* when the sub-clusters of clusters 1, 2 and 3 were calculated. Clusters were obtained with non-transformed data to approximate to normality (see Subsect. 3.2).

In Table 2, column *Main topic* means the most relevant topic according to (20) obtained for the cluster indicated by column *Cluster*; by *Correct #* we mean the correct number of documents in the corpus for the topic in *Main topic*; *Total #* is the number of documents considered to belong to the cluster by our approach; *Act. corr. #* is the number of documents correctly assigned to the cluster; *Prec. (%)* and *Rec. (%)* are Precision and Recall.

Although the original texts of the sub-ELIF corpus are classified by main topic areas, e.g., Agriculture, Energy – Rational use, etc., we have removed that information from the documents before extracting the REs using LocalMaxs, as we wanted to test our approach for clustering usual documents. Although, the topics shown in the previous lists capture the core content of each cluster. However, each cluster is not a perfect "translation" of the "corresponding" clusters in the other languages. Some reasons may help to explain why. Thus, because in

Portuguese we write *Estado-Membro* (just one word) for *Member State*, it will not be in cluster 2, since LocalMaxs does not extract unigrams. Moreover, not every RE has the same number of occurrences of the corresponding RE in the other languages. For example, there are 55 occurrences of *Nomenclatura Combinada*, 54 of *nomenclature combinée*, 45 of *combined nomenclature* and 10 of *Combined Nomenclature*. Then, under the criterion used for extracting topics (see Sect. 4), 45 occurrences is less than 50% of the 108 documents of the cluster 1. Therefore *combined nomenclature* is not considered a topic in cluster 1.

As we keep the original words in text, the same topic may be shown in different forms, e.g., *EUROPEAN COMMUNITIES* and *European Communities*. Though some REs are not topics or just weakly informative expressions, such as *Journal officiel* or *general rules*, we think that about 80% of these REs can be considered correct topics / subtopics, exposing the core content of the clusters.

Since our approach is not oriented for any specific language, we believe that different occurrences for the same concept in the three different languages, are the main reason for different Precision and Recall scores comparing "corresponding" clusters. Furthermore, some documents have REs written in more than one language, for example there is a "Portuguese document" containing some important REs written in French. This reduced Precision of cluster 3 and Recall of cluster 2, since this document was oddly put in cluster 3.

We transformed data to approximate to normality (see Subsect. 3.2) and use it as input to clustering software. Unfortunately, this modified data caused a decrease of Precision (about 20% lower) and the number of subclusters for clusters 1, 2 and 3 were different. However, we will work again on other data transformations with larger corpora in order to obtain better results.

As has been said in Sect. 3.1, the larger the BIC value, the stronger the evidence of clustering (see [4]). Then, BIC can be used to decide whether to "explore" sub-clusters or not. However, we did not work on this problem yet.

6 Related Work

Some known approaches for extracting topics and relevant information use morphosyntactic information, e.g., TIPSTER [7]. So, these approaches would need specific morphosyntactic information in order to extract topics from documents in other languages, and that information might not be available.

In [8], a multidocument summarizer, called MEAD is presented. It generates summaries using cluster topic detection and tracking system. However, in this approach, topics are unigrams. Though many uniwords have precise meanings, multiword topics are usually more informative and specific than unigrams.

In [9], an approach for clustering documents is presented. It is assumed that there exists a set of topics underlying the document collection to cluster, since topics are not extracted from the documents. When the number of clusters is not given to this system, it is calculated based on the number of topics.

7 Conclusions

We presented an unsupervised statistics-based and language independent approach for document clustering and topic extraction. It was applied to a multilingual corpus, using just information extracted from it. Thousands of REs were extracted by LocalMaxs algorithm and then, transformed into a small set of new features, which – according to the results obtained – showed good document *classification* power. The best number of clusters was automatically calculated by MBCA and the results obtained led to a rather precise document clustering. Thus, the number of clusters was not left to the user choice, as it might correspond to an *unnatural* clustering. Although we tested this approach on a small corpus (872,795 words), the results are encouraging, since about 80% of the clusters REs are sufficiently informative for being taken as topics of documents. This lead us to believe that topics, rather than long sentences belonging to just one or two documents, are adequate to define clusters core content.

We tried some data transformations when some distributions suggested lack of normality. However, considering the clustering software we used, the preliminary results showed that the negative effect of these data modifications seems to be more important than the positive effect they produce to make distributions more "normal (Gaussian) looking". Although, further research is required on larger corpora to solve this and other problems.

References

1. Silva, J. F., Dias, G.,Guilloré, S., Lopes, G. P. 1999. Using LocalMaxs Algorithm for the Extraction of Contiguous and Non-contiguous Multiword Lexical Units. *Lectures Notes in Artificial Intelligence, Springer-Verlag*, volume 1695, pages 113-132.
2. Silva, J. F., Lopes, G. P. 1999. A Local Maxima Method and a Fair Dispersion Normalization for Extracting Multiword Units. In *Proceedings of the 6th Meeting on the Mathematics of Language*, pages 369-381, Orlando.
3. Escoufier Y., L'Hermier, H. 1978. A propos de la Comparaison Graphique des Matrices de Variance. *Biometrischc Zeitschrift*, 20, pages 477-483.
4. Fraley, C., Raftery, A. E. 1998. How many clusters? Which clustering method? - Answers via model-based cluster analysis. *The computer Journal*, 41, pages 578-588.
5. Johnson R. A., Wichern, D. W. 1988. *Applied Multivariate Statistical Analysis, second edition.* Prentice-Hall.
6. Box, G. E. P., D. R. Cox. 1964. *An Analysis of Transformations*, (with discussion). Journal of the Royal Statistical Society (B), 26, no. 2, pages 211-252.
7. Wilks, Y., Gaizauskas, R. 1999. Lasie Jumps the Gate. In Tomek Strzalkowski, editor, *Natural Language Information Retrieval.* Kluwer Academic Publishers, pages 200-214.
8. Radev, D. R., Hongyan, J., Makgorzata, B. 2000. *Centroid-based summarization of multiple documents: sentence extraction, utility-based evaluation, and user studies.* Proceedings of the ANLP/NAACL Workshop on Summarization.
9. Ando, R. K., Lee L. 2001. *Iterative Residual Rescaling: An Analysis and Generalization of LSI.* To appear in the proceedings of SIGIR 2001.

Sampling-Based Relative Landmarks: Systematically Test-Driving Algorithms before Choosing

Carlos Soares[1], Johann Petrak[2], and Pavel Brazdil[1]

[1] LIACC/FEP, University of Porto {csoares,pbrazdil}@liacc.up.pt
[2] Austrian Research Institute for Artificial Intelligence johann@ai.univie.ac.at

Abstract. When facing the need to select the most appropriate algorithm to apply on a new data set, data analysts often follow an approach which can be related to test-driving cars to decide which one to buy: apply the algorithms on a sample of the data to quickly obtain rough estimates of their performance. These estimates are used to select one or a few of those algorithms to be tried out on the full data set. We describe *sampling-based landmarks* (SL), a systematization of this approach, building on earlier work on landmarking and sampling. SL are estimates of the performance of algorithms on a small sample of the data that are used as predictors of the performance of those algorithms on the full set. We also describe *relative landmarks* (RL), that address the inability of earlier landmarks to assess relative performance of algorithms. RL aggregate landmarks to obtain predictors of relative performance. Our experiments indicate that the combination of these two improvements, which we call Sampling-based Relative Landmarks, are better for ranking than traditional data characterization measures.

1 Introduction

With the growing number of algorithms available for data mining, the selection of the one that will obtain the most useful model from a given set of data is an increasingly important task. Cross-validation with adequate statistical comparisons is currently the most reliable method for algorithm selection. However, with the increasing size of the data to be analyzed, it is impractical to try all available alternatives.

The most common approach to this problem is probably the choice of the algorithm based on the experience of the user on previous problems with similar characteristics. One drawback with this approach is that it is difficult for the user to be knowledgeable about all algorithms. Furthermore, two sets of data may look similar to the user but they can be sufficiently different to cause significant changes in algorithm performance.

In meta-learning, the idea of using knowledge about the performance of algorithms on previously processed data sets is systematized [1,5]. A database of meta-data, i.e., information about the performance of algorithms on data sets and characteristics of those data sets, is used to provide advice for algorithm selection, given a new problem. Most approaches to meta-learning have concentrated on selecting one or a set of algorithms whose performance is expected to be the best [1,12]. However, taking into account that no prediction model is perfect and that in most meta-learning settings the amount of meta-data is not very large, there is a growing interest in methods that provide a ranking of the algorithms according to their expected performance [3,9,14]. The ranking approach is more flexible, enabling the user to try out more than one algorithm depending on the available resources as well as to accommodate his/her personal preferences. Probably the most important issue in meta-learning is how to obtain good data characteristics, i.e. measures that provide information about the performance of algorithms and that can be computed fast. The earlier approaches used three types of measures, general (e.g. number of examples), statistical (e.g. number of outliers) and information-theoretic (e.g. class entropy) [10, sec. 7.3]. Landmarks were recently suggested as data characterization measures [2,12]. A landmark is a simple and efficient algorithm that provides information about the performance of more complex algorithms. For example, the top node in a decision tree can be an indicator of the performance of the full tree.

Another common approach to the problem of algorithm selection is sampling, which consists of drawing a (random) sample from the data and trying at least a few of the available algorithms on it. The algorithm performing best on this sample is then applied to the full data set. Petrak [11] performed a systematic study of this approach in the supervised classification setting. Even with simple sampling strategies, positive results were obtained on the task of single algorithm selection. However, the learning curves presented in that work indicate that large samples are required to obtain a clear picture of the relative performance between the algorithms. In this paper we combine both ideas, landmarks and sampling, in a framework for data characterization, called *sampling-based landmarks* (SL). In SL the results of algorithms on small samples are not used directly as estimators of algorithm performance, as it is done in sampling. They serve as predictors in the same spirit of landmarking, i.e. as data characteristics used in the meta-learning process.

Although landmarking has also shown positive results for single algorithm selection [2,12], it does not provide much information about the relative performance of algorithms, as recent results show [3]. To address this problem, we use *relative landmarks* (RL). RL aggregate landmarks to obtain measures that are predictors of relative, rather than individual, performance. We believe they will be more appropriate for ranking and algorithm selection in general than traditional (non-relative) landmarks. The experiments performed in this paper combine these two improvements to obtain Sampling-based Relative Landmarks.

When providing support to data mining users, one must take into account that the evaluation of the model is not only based on its accuracy. Here we will work in the setting defined in [14], where the accuracy and total execution time of supervised classification algorithms are combined.

In the next section we present sampling-based landmarks. In Section 3 we then describe relative landmarks and how to implement them using the Adjusted Ratio of Ratios, which is a multicriteria evaluation measure for classification algorithms. In Section 4 we combine both concepts into sampling-based relative landmarks and compare them empirically with traditional data characterization measures for ranking. Related work is discussed in Section 5 and in the final section we present some conclusions and describe future work.

2 Sampling-Based Landmarks

Landmarks were introduced in [2,12] as fast and simple learning algorithms whose performance characteristics can be used to predict the performance of complex algorithms. The simplicity of the chosen algorithms was necessary in order to achieve acceptable speed: meta-learning will only be an interesting alternative to cross-validation if data set characterization is significantly faster than running the algorithms. As an alternative to simplifying algorithms, we can achieve speed by reducing the size of the data. A *Sampling-based landmark* is an estimate of the performance of an algorithm obtained by testing a model that was obtained with that algorithm on a small sample of a data set on another small sample of the same data.

Different sampling strategies have been described in the literature to get good performance estimates by using only a small amount of data. Simple strategies include sampling a fixed number or a fixed fraction of cases. Other strategies try to estimate the number of cases needed by analyzing certain statistical properties of the sample [7,8]. Still more elaborate strategies try to find an optimum sample size by repeated evaluation of increasing samples [13].

Here, we use the simple approach of randomly selecting a constant size sample of 100 cases, and a fixed fraction of 10% for testing. Thus, the (usually critical) training effort is reduced to constant time complexity, while testing time is reduced by a constant factor (sampling itself is comparably fast with linear time complexity for sequential access files). Furthermore, the results presented in [11] indicate that the quality of the estimates depend more on the size of the test sample than on the size of the training sample.

The use of Sampling-based Landmarks is not restricted to predicting accuracy. For instance, the time to obtain the landmark can also be used to predict the time to run the algorithm on the full set of data and the same applies to the complexity of the model. As explained below, here we concentrate on accuracy and time.

3 Relative Landmarks

Landmarks can be generally useful as indicators of algorithm performance. However, they do not provide information about the *relative* performance of algorithms, as is required for algorithm selection and, in particular, for ranking [3]. To provide relative performance information, landmarks should be aggregated in some way, leading to the notion of *relative landmarks* (RL). Here RL are based on the Adjusted Ratio of Ratios (ARR), a multicriteria evaluation measure combining accuracy and time to assess relative performance [14]. In ARR the compromise between the two criteria involved is given by the user in the form "the amount of accuracy I'm willing to trade for a 10 times speed-up is $X\%$". The ARR of algorithm i when compared to algorithm j on data set d is defined as follows:

$$ARR_{i,j}^{d} = \frac{\frac{A_i^d}{A_j^d}}{1 + \log\left(\frac{T_i^d}{T_j^d}\right) * X}$$

where A_i^d and T_i^d are the accuracy and time of i on d, respectively. Assuming, without loss of generality, that algorithm i is more accurate but slower than algorithm j on data set d, the value of $ARR_{i,j}^{d}$ will be 1 if i is $X\%$ more accurate than j for each order of magnitude that it is slower than j. $ARR_{i,j}^{d}$ will be larger than one if the advantage in accuracy of i is $X\%$ for each additional order of magnitude in execution time and vice-versa.

When more than two algorithms are involved, we can generate a relative landmark for each of the n landmarks, l, based on ARR in the following way:

$$rl_i^d = \frac{\sum_{j \neq i} ARR_{i,j}^{d}}{n-1} \qquad (1)$$

Each relative landmark, rl_i^d, represents the relative performance of landmark i when compared to all other landmarks on data set d[1]. Therefore, a set of relative landmarks, rl_i^d, can be used directly as meta-features, i.e. characteristics of the data set, for ranking. Given that the main purpose of relative landmarks is characterizing the relative performance of the algorithms, we can alternatively order the values of the relative landmarks, and use the corresponding ranks, $r_{rl^d}(i)$, as meta-features.

4 Experiments with Sampling-Based Relative Landmarks

The aim of our experiments was to compare data set characterization by traditional measures (general, statistical and information-theoretic) and by sampling-based relative landmarks (SRL), i.e. a combination of both improvements introduced above. We will refer to the SRL as SRL-scores, their ranks as SRL-ranks and the traditional data characterization as DC-GSI.

Next we will describe the experimental setting used to obtain the meta-data. Then, we describe the meta-learning methods used and, finally, the results.

Meta-Data: SRL are obtained very simply by creating relative landmarks (Section 3) using the performance of the candidate algorithms on a sample of

[1] Since the relative landmarks, rl_i^d, depend on the X parameter and on the set of algorithms used, we store raw performance information, A_i^d and T_i^d and calculate the relative landmarks on-the-fly.

the data (Section 2). Assuming that $s(d)$ is the sampling method used, n is the number of algorithms, we calculate one SRL, $rl_i^{s(d)}$, for each algorithm i using Eq. 1, thus obtaining a set of n SRL measures. Here we have used a simple sampling strategy: train on a stratified random sample with 100 cases and test on 10% of the rest of the data set.

We have used 45 data sets with more than 1000 cases, mostly from the UCI repository [4] but also a few others that are being used on the METAL project[2] (SwissLife's Sisyphus data and a few applications provided by DaimlerChrysler). Ten algorithms were executed on these data sets[3]: two decision tree classifiers, C5.0 and Ltree, which is a decision tree that can introduce oblique decision surfaces; the IB1 instance-based and the naive Bayes classifiers from the MLC++ library; a local implementation of the multivariate linear discriminant; two neural networks from the SPSS Clementine package (Multilayer Perceptron and Radial Basis Function Network); two rule-based systems, C5.0 rules and RIPPER; and an ensemble method, boosted C5.0. The algorithms were all executed with default parameters. Since we have ten algorithms, we generated a set of ten SRL measures and another with the corresponding ranks for each data set. In the DC-GSI setting we have used the 25 meta-features that have performed well before [14].

Meta-learning Methods: Two ranking methods were used. The first is the Nearest-Neighbor ranking using ARR as multicriteria performance measure [14]. Three scenarios for the compromise between accuracy and time were considered, $X \in \{0.1\%, 1\%, 10\%\}$, for increasing importance of time. To obtain an overall picture of the performance of the ranking method with the three different sets of data characterization measures, we varied the number of neighbors (k) from 1 to 25. The second meta-learning method consists of building the ranking directly from the sampling-based relative landmarks. In other words, the recommended ranking for data set d is defined by the corresponding SL-ranks, $r_{rl^{s(d)}}(i)$. This method can be regarded as a baseline that measures the improvement obtained by using sampling-based relative landmarks as predictors, i.e. as meta-features used by the NN ranking method, rather than directly as estimators for ranking. This method is referred to as SL-ranking.

The evaluation methodology consists of comparing the ranking predicted by each of the methods with the estimated true ranking, i.e. the ranking based on the estimates of the performance of the algorithms on the full set of data [14]. The distance measure used is average Spearman's rank correlation that yields results between 1 (for perfect match) and -1 (inverted rankings). The experimental methodology is leave-one-out.

Results: The results are presented in Figure 1. The first observation is that in all scenarios of the accuracy/time compromise, SL-ranking always achieves worse results than the alternatives. This is consistent with an observation that can be made from previous work on sampling [11], namely that relatively large sample

[2] Esprit Long-Term Research Project (#26357) *A Meta-Learning Assistant for Providing User Support in Data Mining and Machine Learning* (www.metal-kdd.org).

[3] References for these algorithms can be found in [11].

sizes are required to correctly rank the algorithms directly from the performance on the sample.

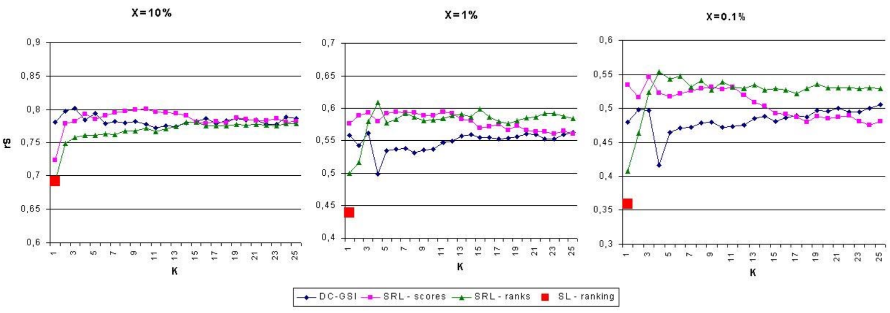


Fig. 1. Average Spearman rank correlation coefficient (r_S) obtained by NN/ARR ranking using two types of sampling-based relative landmarks (SRL-scores and SRL-ranks) and traditional data characterization (DC-GSI) for varying number of neighbors (k) in three different scenarios of the compromise accuracy/time. Also included is the performance of the ranking obtained directly from the sampling-based relative landmarks (SL-ranking).

These results also indicate that SRL-scores represent the best overall method especially for reasonable number of neighbors ($k < 10$, which represents roughly 20% or less of the number of data sets), except when time is the dominant criterion. In the latter case and for $k < 4$, DC-GSI performs better. This is expected because for such a small sample size, the precision (number of decimal places) used to measure time is not sufficient to be discriminative. However, it is interesting to note that in the same scenario, SRL-scores is again comparable or better than DC-GSI for $4 \leq k \leq 10$.

Focussing the comparison on both types of SRL, SRL-scores and SRL-ranks, we observe that the former seem to be much more stable with smaller number of neighbors. We believe that this makes this approach more appealing, although in the scenario where accuracy is the dominant criterion, the performance obtained with SRL-ranks is better for $K \in \{4, 5, 6\}$. There is also a tendency for SRL-ranks to be better than SRL-scores for larger, less common, number of neighbors ($10 \leq k \leq 25$). These results are somewhat surprising, given that SRL-scores are expected to be more sensitive to outliers. We would expect to mitigate this problem by using the information about the performance of algorithms at a more coarse level, i.e. SL-ranks. However, we seem to lose too much information with the latter.

5 Related Work

The concepts of relative landmarks and subsampling landmarks, which are essentially the same as the ones described here, have been presented by Fürnkranz *et al* [6]. The authors also present an interesting taxonomy of relative landmarks, that includes both types used here, i.e. using scores or their ranks. However, there are two main differences between the two approaches, the most important being that we combine both concepts, obtaining sampling-based relative landmarks. Also, in [6] the aim is to select a single algorithm while we are concerned with ranking the candidate algorithms.

6 Conclusions

Landmarks have been introduced in [2,12] as simple and efficient algorithms that are useful in predicting the performance of more complex algorithms. We observe that efficiency can be obtained not only by simplifying algorithms but also by sampling the data. *Sampling-based landmarks* are obtained by running the candidate algorithms on small samples of the data and are used as data set characterization measures rather than as estimators of algorithm performance [11]. As landmarks have little ability to predict relative performance, this makes them less suitable for ranking, as it was shown empirically [3]. *Relative landmarks*, obtained by adequately aggregating landmarks, can be used to address this issue. In this paper we have used the Adjusted Ratio of Ratios [14], a multicriteria relative performance measure that takes accuracy and time into account, for that purpose.

We have empirically evaluated the combination of these two improvements to landmarking, referred to as sampling-based relative landmarks. Here, they were based on very small samples. We compared them to a set of traditional data characterization measures and also to rankings generated directly from the sampling-based landmarks. In our experiments with a Nearest-Neighbor ranking system that also used the ARR multicriteria measure, the best results in general were obtained with sampling-based relative landmarks when the number of neighbors ranged from 10% to 20% of the total number of data sets.

Although the results are good, there is still plenty of work to do. Concerning sampling-based landmarks, we intend to study the effect of increasing sample size and other sampling methods. We will also try to reduce the variance of the estimates and to generate simple estimators of the learning curves of each algorithms. We plan to combine sampling-based landmarks with a selection of traditional characterization measures. We also intend to improve relative landmarks, namely by analyzing other ways of aggregating performance and to apply relative landmarks to traditional landmarks, i.e. obtained by simplifying algorithms. Finally, we note that both frameworks, sampling-based and relative landmarks, can be applied in settings where multicriteria performance measures other than ARR are used [9] as well as settings where accuracy is the only important performance criterion.

References

1. D.W. Aha. Generalizing from case studies: A case study. In D. Sleeman and P. Edwards, editors, *Proceedings of the Ninth International Workshop on Machine Learning (ML92)*, pages 1–10. Morgan Kaufmann, 1992.
2. H. Bensusan and C. Giraud-Carrier. Casa batló is in passeig de grácia or landmarking the expertise space. In J. Keller and C. Giraud-Carrier, editors, *Meta-Learning: Building Automatic Advice Strategies for Model Selection and Method Combination*, pages 29–46, 2000.
3. H. Bensusan and A. Kalousis. Estimating the predictive accuracy of a classifier. In P. Flach and L. de Raedt, editors, *Proceedings of the 12th European Conference on Machine Learning*, pages 25–36. Springer, 2001.
4. C. Blake, E. Keogh, and C.J. Merz. Repository of machine learning databases, 1998.
http:/www.ics.uci.edu/∼mlearn/MLRepository.html.
5. P. Brazdil, J. Gama, and B. Henery. Characterizing the applicability of classification algorithms using meta-level learning. In F. Bergadano and L. de Raedt, editors, *Proceedings of the European Conference on Machine Learning (ECML-94)*, pages 83–102. Springer-Verlag, 1994.
6. J. Fürnkranz and J. Petrak. An evaluation of landmarking variants. In C. Giraud-Carrier, N. Lavrac, and S. Moyle, editors, *Working Notes of the ECML/PKDD 2000 Workshop on Integrating Aspects of Data Mining, Decision Support and Meta-Learning*, pages 57–68, 2001.
7. B. Gu, B. Liu, F. Hu, and H. Liu. Efficiently determine the starting sample size for progressive sampling. In P. Flach and L. de Raedt, editors, *Proceedings of the 12th European Conference on Machine Learning*. Springer, 2001.
8. G.H. John and P. Langley. Static versus dynamic sampling for data mining. In E. Simoudis, J. Han, and U. Fayyad, editors, *Proceedings of Second International Conference on Knowledge Discovery and Data Mining (KDD-96)*. AAAI-Press, 1996.
9. J. Keller, I. Paterson, and H. Berrer. An integrated concept for multi-criteria ranking of data-mining algorithms. In J. Keller and C. Giraud-Carrier, editors, *Meta-Learning: Building Automatic Advice Strategies for Model Selection and Method Combination*, 2000.
10. D. Michie, D.J. Spiegelhalter, and C.C. Taylor. *Machine Learning, Neural and Statistical Classification.* Ellis Horwood, 1994.
11. J. Petrak. Fast subsampling performance estimates for classification algorithm selection. In J. Keller and C. Giraud-Carrier, editors, *Meta-Learning: Building Automatic Advice Strategies for Model Selection and Method Combination*, pages 3–14, 2000.
12. B. Pfahringer, H. Bensusan, and C. Giraud-Carrier. Tell me who can learn you and i can tell you who you are: Landmarking various learning algorithms. In P. Langley, editor, *Proceedings of the Seventeenth International Conference on Machine Learning (ICML2000)*, pages 743–750. Morgan Kaufmann, 2000.
13. F. Provost, D. Jensen, and T. Oates. Efficient progressive sampling. In S. Chaudhuri and D. Madigan, editors, *Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.* ACM, 1999.
14. C. Soares and P. Brazdil. Zoomed ranking: Selection of classification algorithms based on relevant performance information. In D.A. Zighed, J. Komorowski, and J. Zytkow, editors, *Proceedings of the Fourth European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD2000)*, pages 126–135. Springer, 2000.

Recursive Adaptive ECOC models

Elizabeth Tapia[1], José Carlos González[1], Javier García-Villalba[2], Julio Villena[1]

[1]Department of Telematics Engineering - Technical University of Madrid, Spain
{etapia, jgonzalez, jvillena}@gsi.dit.upm.es
[2]Department of Informatic Systems, Complutense University of Madrid, Spain
javiergv@sip.ucm.es

Abstract. A general framework for the design of error adaptive learning algorithms in multiple output domains based on Dietterich's ECOC approach, recursive error output correcting codes and iterative APP decoding methods is proposed. A particular class of these Recursive ECOC (RECOC) learning algorithms based on Low Density Parity Check is presented.

1 Introduction

As the ECOC [1] decoding strategy is roughly of a hard decision type, the final hypothesis cannot exploit the knowledge arising from the observed binary learners' performances. In this way, the ECOC approach cannot lead to any error adaptive learning algorithm except when used in the core of AdaBoost variants [2]. This is a striking quest considering that original AdaBoost formulation is essentially binary oriented. Furthermore, both AdaBoost and ECOC seem to be different for Machine Learning literature [3], even when ECOC should be natural extension of AdaBoost. This work follows the approach introduced in [4][5]. In this paper, a particular instance of RECOC learning based Low-Density Parity Check (LDPC) codes [6] is presented.

The remainder of this paper is organized as follows. In section 2, we review the RECOC approach. In section 3, we present the RECOC_LDPC learning algorithm. In section 4, we present experimental results. Finally, in Section 5 conclusions and further work are presented.

2 Low Complexity Error Adaptive Learning Algorithms

In previous work [4], we demonstrated that AdaBoost decision could be interpreted as an instance of threshold decoding [7] for a T-repetition[1] code under the assumption of a binary discrete memoryless channel constructed by the ensemble of binary weak

[1] Each codeword of length T carries a unique information bit and T-1 replicas. The unique *informative codeword bit* can be recovered by simple majority or threshold decoding.

learners. Let us consider the application of this result to the design of a generalized class of error adaptive ECOC algorithms based on binary error correcting codes. As a first example, we can think about the class of ECOC algorithms induced by binary linear codes. Each M-valued output label can be broken down into k , $k \geq \lceil \log_2 M \rceil$ binary labels by means of a suitable quantization Q_M process. Afterwards, each k −bitstream can be channel encoded using codewords of length $n \succ k$. Following the ECOC approach, a noisy codeword $\mathbf{h}$ can be computed at the learner's side so that the learning problem can be expressed as a general decoding instance. Decoding is in essence a decision making process. Given a received noisy codeword, a decoder tries to figure out a transmitted information bit or a transmitted codeword. *Maximum Likelihood* decoding while minimizing the probability of codeword error do not necessary minimizes the probability of symbol (bit) error. Symbol *Maximum A Posteriori* (MAP) decoding methods effectively minimizes bit error probability by computing the maximum on the set of *a posteriori probabilities* (APP) [7]. Let c_i be a codeword bit in a codeword $\mathbf{c}$ of length n and k informative codeword bits at the first k positions. The APP probabilities are as follows

$$p(c_i \mid context) \quad 1 \leq i \leq k \qquad \textbf{(1)}$$

The event *context* refers in general to all available information at symbol decoding time including channel statistics, the received noisy codeword $\mathbf{h}$ and the mathematical description of the coding scheme. For linear block codes such structure is defined by the parity check matrix H or the generator matrix G^{T} .

Definition (linear block code) A linear (n,k) block code is defined by a $n \times k$ binary generator matrix $\mathbf{G}$ so that a k -bit binary source message $\mathbf{u}$ is encoded as a n − bit vector or codeword $\mathbf{c} = \mathbf{u} \cdot \mathbf{G}$ mod 2. In addition for a systematic code $G = [\mathbf{I}_k \quad \mathbf{P}]$. Alternatively, a linear block code can be defined through parity check matrix $H_{(n-k)\times n}$ for which $H_{(n-k)\times n}$ the relation $H \cdot \mathbf{c}^* = \mathbf{0}$ mod 2 holds, being $\mathbf{c}^*$ the transpose codeword.

In our learning-decoding framework, channel statistics are defined by statistical behavior of binary learners' errors. In this way, we can think about a discrete memoryless channel constructed in a training phase and explicitly used in future predictions by means of suitable APP decoding algorithms. The use of APP decoding techniques for ECOC expansions based on binary linear codes allows the design of error adaptive learning algorithms in arbitrary output domains. It should be noted, however, that error adaptation is only one of the learning constraints. It is well known that ECOC schemes should resemble random coding in order to achieve successful learning. ECOC codes found by exhaustive search cannot be decoded by APP methods as no underlying mathematical structure can be defined on them. It is worthwhile to note that the ECOC formulation though requiring almost pseudorandom codes does not take advantage of the random coding characteristic in the decoding step. To be fair, one should remember that practical coding theory did the same for almost 50 years until the development of iterative decoding techniques [8] for recursive error correcting codes [9] like Turbo Codes [10][11] or LDPC codes.

These binary linear codes are for construction inherently pseudorandom. Decoding is performed in a recursive way each using time APP decoding methods. Regarding these considerations, we will focus our attention on ECOC alike algorithms based binary linear recursive error correcting codes and iterative APP decoding techniques i.e. the so called on RECOC models [5].

By means of a simple example, let us explain how the above considerations fit in the design of low complexity error adaptive learning algorithms in non-binary output domains. Let us assume $M = 16$ so that $k = 4$ can be assumed. Each output label is first expressed in bitstreams $\mathbf{u} = (u_0,\ u_1,\ u_2,\ u_3)$ and then channel encoded by means of a simple $(8,4,4)$[2] extended Θ Hamming code. The code is systematic with sixteen different codewords. A transmitted $\mathbf{c} = (u_0,\ u_1,\ u_2,\ u_3,\ z_4,\ z_5,\ z_6,\ z_7)$ Θ -codeword has parity bits z_i, $4 \le i \le 7$ defined as follows

$$\begin{aligned} z_4 + u_0 + u_1 + u_2 &= 0 \\ z_5 + u_0 + u_1 + u_3 &= 0 \\ z_5 + u_0 + u_2 + u_3 &= 0 \\ z_5 + u_1 + u_2 + u_3 &= 0 \end{aligned} \qquad \textbf{(2)}$$

Each equation in **(3)** can be considered a simple parity check code for the transmission of *three* information bits. In learning terms, our $(8,4,4)$ code can be also defined in terms of *four* parity check subcodes, each of them involved in the learning of some target concept in an M'=8 output space. This recursive view can be modeled by means of the so-called Tanner graphs [9][12]. The set of parity coding constraints S_t, $0 \le t \le 3$, and codeword bits define the nodes in the graph while edges are put connecting parity constraints to codeword bits (see **Fig. 1**).

Redundant concepts are introduced in order to cope with the underlying learning noise. A received noisy codeword $\mathbf{h} = (h_0,\ h_1,\ h_2,\ h_3,\ h_4,\ h_5,\ h_6,\ h_7)$ is the result of a random toggling process on codeword bits (binary concepts) in $\mathbf{c} = [c_i]$, $0 \le i \le 7$. A simple model for this toggling process is a binary discrete additive one.

$$f_i(c_i) = P(h_i \mid c_i) = \begin{cases} p_i & h_i \ne c_i \\ 1 - p_i & h_i = c_i \end{cases} \qquad 0 \le i \le 7 \qquad \textbf{(3)}$$

.In this way, the above set of constraints can be directly introduced in a Tanner graph. h_i variables are instantiated at codeword reception time as they are the binary learners' predictions. In addition, information bits (binary target concepts defining the M-valued one) can be constrained by a prior distribution $g_i = p(u_i)$, $0 \le i \le 3$. The resulting extended Tanner graph is known as the code Factor graph [8] (**Fig. 1**). A RECOC rule requires the computation of probabilities $p(c_i \mid \Theta, f_o, ..., f_{n-1}, g_o, ... g_{k-1})$, so that we can give an estimate c_i * of each binary informative target concept

[2] Using standard coding notation, it denotes a $(n\ ,k\ ,d\ )$ linear block code, being n the codeword length, k the number of information bits and d the minimum code distance

$$c_i^* = \arg \underset{u_i}{Max} \ \left[p(c_i \mid \Theta, f_0,..,f_{n-1}, g_0,...g_{k-1})\right] \quad i = 0,....,k-1 \tag{4}$$

These probabilities can be computed by a message-passing algorithm namely a generalized version of Pearl's Belief propagation [13] algorithm in graph with cycles. The observed results in the coding field are almost optimal, despite of the presence of cycles. The only requirement would be knowledge of probabilities p_i, $0 \le i \le n-1$, governing the learning noise. However, these probabilities can be estimated from the training error responses achieved by the set of binary learners under the training sets TS_i induced in encoding ECOC training phase from the original training sample TS .

$$p_i = P_{TS_i}\left[h_i(\mathbf{x}) \neq y\right] \qquad 0 \le i \le n-1 \tag{5}$$

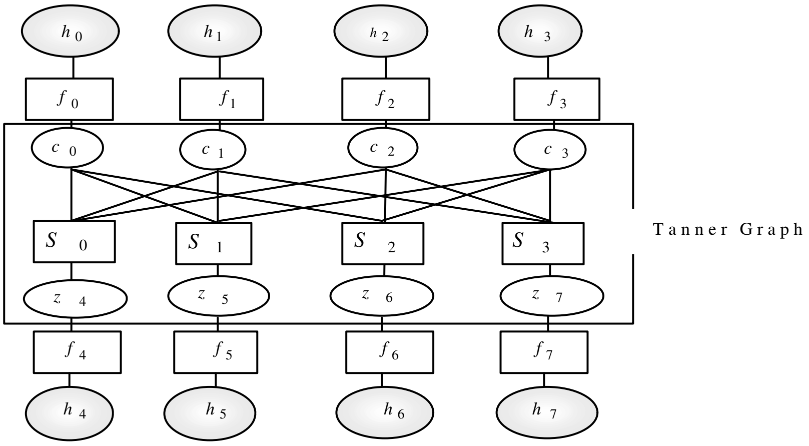


Fig. 1. Factor Graph for the (8,4,4) extended Hamming code

3 RECOC_LDPC

In order to obtain good results with iterative decoding techniques applied to recursive codes based on simple parity check codes, some additional constraints on the Tanner graph structure are required. These constraints are met by the so-called LDPC codes.

Definition LDPC codes [6]. A Low-Density Parity Check (LDPC) code is specified by a pseudorandom parity check matrix H containing mostly 0´s and a small number of 1´s. A binary (n, j, k) LDPC code has block length n and a parity-check matrix H with exact j ones per column and k 1´s in each row, assuming $j \ge 3$ and $k \ge j$. Thus every code bit is checked by exactly j parity checks and every parity check involves k codeword bits. The typical minimum distance of these codes increases linearly with n for a fixed k and j .

Let us consider the LDPC code shown in **Fig. 2** through its factor graph representation. Such a code is constructed from five parity check subcodes thus giving parity coding constraints $S_j, 0 \le j \le 4$. LDPC codes can be decoded a generalization of Pearl's Belief Propagation algorithm [14][15]. Component subcodes i.e. parity constraints behave like communication sockets for the set of involved binary learners WL_i issuing the observed binary predictions h_i on binary target concepts c_i, $0 \le i \le 9$. It should be noted that the BP algorithm is concerned with computation of conditional probabilities over cycle-free graphs and factor graph models are far away from being cycle-free. However, iterative decoding algorithms on Tanner graphs with cycles based on the BP algorithm work almost optimally. Therefore, they may also perform well for our RECOC_LDPC learning algorithm.

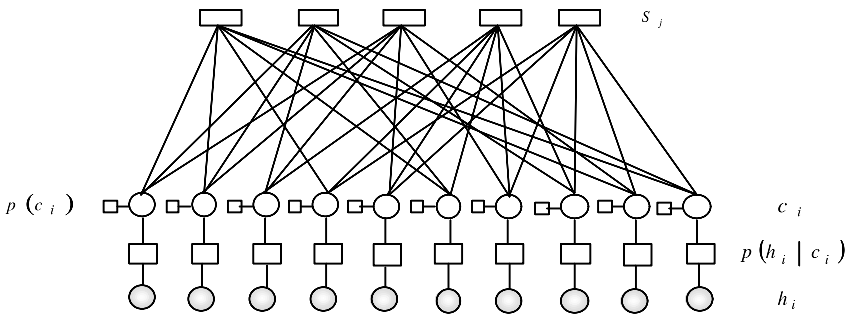


Fig. 2. A (n=10, j=3, k=5) LDPC code mapped into a RECOC_LDPC scheme

```
RECOC_LDPC Algorithm
Input
(n, j, k) LDPC by the parity check matrix $H$ and $G$
Quantizer $Q_M$ with $k$ bits per input in $\{1,...,M\}$
Training Sample $TS$ with $|TS| = m_{TS}$, Distribution $D$ on $TS$
Binary Weak Learner $WL$, Iterations I for BP
Processing
  RECOC ($TS$, $Q_M$, $G$, $WL_0$, . . ., $WL_{n-1}$, $p_o$, . . . , $p_{n-1}$))
Output
  $h_f(\mathbf{x}) = Q_M^{-1}(BP(H, \mathbf{x}, I, WL_0, . . . , WL_{n-1}, p_0, . . . , p_{n-1}))$
```

The RECOC procedure performs the ECOC encoding stage but with additional computation of training error responses $p_i, 0 \le i \le n-1$. In the prediction stage, this set will be used in the BP decoding with **I** of iterative decoding steps. Then, a set of k informative binary target concepts will be estimated. Finally, a final hypothesis will be obtained by application of inverse quantization process Q_M^{-1}.

4 Experimental Results for RECOC_LDPC learning

We have tested the RECOC_LDCP algorithm on various UCI learning domains. Learning algorithms were developed using public domain Java WEKA library. Therefore, AdaBoost (AB), Decision Stump[3] (DS) and C4.5[4] (denoted by C4 for the multiclass case) implementations details can be fully determined from WEKA documentation. For LDPC code creation and the BP propagation algorithm, we developed a library extension of WEKA [16] software using public domain D. J. MacKay software [15]. We adopted $(n,4,k)$ LDPC output encoding schemes with $n=\frac{\lceil \log_2 M \rceil}{R}$, being $R=1-\frac{k}{j}$ the channel rate in number of information bits (informative binary target concepts) per codeword length n. Let us denote by RECOC_LDCP(R) + DS, the test error rate for RECOC_LDPC learning at channel rate R using DS learners. Also, let us denote by RECOC_LDCP(R) + AB(T) + DS the test error for RECOC_LDPC learning at channel rate R with T inner binary AdaBoost boosting steps on DS learners. Test error responses for learning domains, Anneal, Primary Tumor, Lymph and Audiology against a maximum number of iterations **I** used in the BP algorithm are shown from **Fig. 3** to **Fig. 6**. Test error responses without channel encoding i.e. a bare transmission of the quantization scheme Q_M on the M valued output domain are shown for **I** =0.

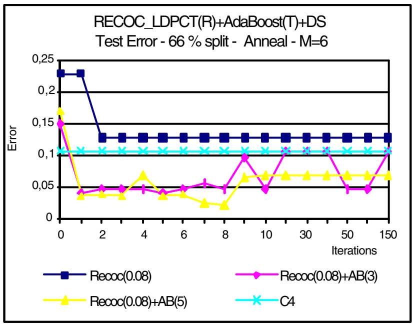


Fig. 3. Anneal. RECOC_LDPC allows test error below C4.

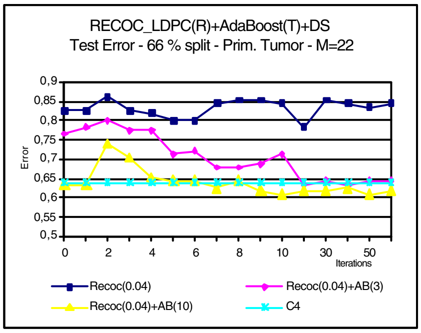


Fig. 4. Primary Tumor. RECOC_LDPC allows test error below C4

[3] Decision tree with only one node

[4] R. Quinlan's decision tree algorithm in its C4.5 Revision 8 public version

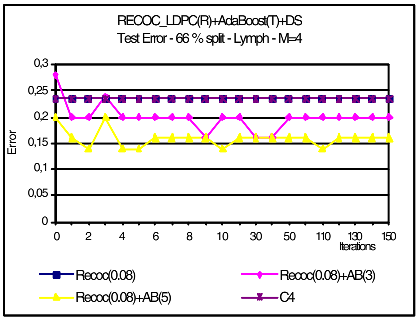


Fig. 5. Lymph. RECOC_LDPC with inner boosting allows test error below C4

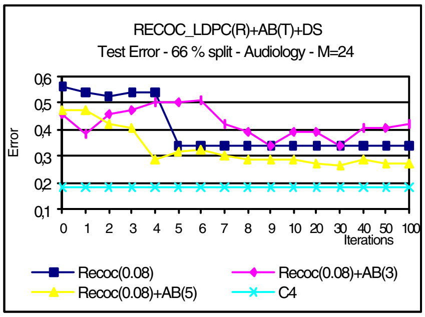


Fig. 6. Audiology. RECOC_LDPC cannot learn below C4.

The observed results show that aside from C4.5 benchmark comparisons, in all cases the desired boosting effect is achieved with respect to the base learning model (DS learners). The observed results are consistent with the typical BP recursive decoding performance for LDPC codes, i.e. better performance for increasing number of iterations steps and decreasing channel rates (not shown). In addition, performance results suggest that for cases where RECOC_LDPC has an error floor (Audiology), the problem might be on the underlying code itself i.e. a representation problem might be causing such undesirable behavior.

5 Conclusions and Further Work

The main contribution of this work has been the development a general decoding framework for the design of low-complexity error adaptive learning algorithms. From this decoding view of the learning from examples problem, a channel is constructed in the training phase and it is later used in a prediction stage. Learning algorithms in this model play a decoding function by means of APP decoding methods under the assumption that a Recursive Error Correcting Output enCoding (RECOC) expansion has taken place at a transmitter or teacher side. This framework embodies the standard AdaBoost algorithm as an instance of threshold decoding for simple repetition codes. RECOC learning models can be described by coding related graphical models. In addition, RECOC predictions can be explained by general message-passage schemes on their underlying graphical models. This is an appealing description of learning algorithms, which allows an intuitive but powerful design approach. A number of directions for further work and research stand out. Besides its validation with real data, it remains to study convergence characteristics, the effect of channel model assumptions and alternative binary learning schemes. Good alternatives for further research are Gallager's algorithms A and B [6] and a simplified implementation [17] of the BP algorithm.

Acknowledgements

Elizabeth Tapia's work is supported by an Argentinean Government FOMEC grant and the National University of Rosario, Argentina. Javier García's work is supported by the Amo Complutense Program and by the Spanish Ministry of Science and Technology (MCYT, Spain) under projects TIC2000-0735 and TIC2000-0737-C03-02. During this work, he was with the IBM Research Division K65/C2 at IBM Almaden Research Center, San Jose, California, USA (javiervi@almaden.ibm.com).

References

1. Dietterich, T., Bakiri, G.: Error-correcting output codes: A general method for improving multiclass inductive learning programs. Proceedings of the Ninth National Conference on Artificial Intelligence (AAAI-91), pp. 572-577, Anaheim, CA: AAAI Press (1991)
2. Schapire, R. E., Singer, Y.: Improved Boosting Algorithms Using Confidence - rated Predictions. Machine Learning, Vol. 37, No. 3, pp. 277-296 (1999)
3. Dietterich, T.: Ensemble Methods in Machine Learning. In J. Kittler and F. Roli (Eds.). Multiple Classifier Systems, MCS 01. Springer Lecture Notes in Computer Science, pp. 1-15, (2000)
4. Tapia, E., González, J.C., Villena, J.: A Generalized Class of Boosting Algorithms Based on Recursive Decoding Models. In J. Kittler and F. Roli (Eds.). Multiple Classifier Systems, MCS 02. Springer Lecture Notes in Computer Science, pp. 22-31, Cambridge, UK, (2001)
5. Tapia, E, González, J.C., Villalba, L: On the design of Error Adaptive ECOC algorithms. In Proc. IDDM 2001 Workshop, 12th ECML, pp. 139-150, Freiburg (2001)
6. Gallager, R. G.: Low Density Parity-Check Codes. Cambridge, Massachusetts, M.I.T. Press (1963)
7. Massey, J.: Threshold Decoding. Cambridge, MA: M.I.T. Press (1963)
8. Kschischang F., Frey, B.: Iterative decoding of compound codes by probability propagation in graphical models. IEEE J. Sel. Areas in Comm. Vol. 16, No. 2, pp. 219-230 (1998)
9. Tanner, M.: A recursive approach to Low Complexity Error Correcting Codes. IEEE Trans. Inf. Theory, Vol. 27, pp. 533-547 (1981)
10. Berrou, C., Glavieux, A.: Near Optimum Error Correcting Coding and Decoding: Turbo Codes. IEEE Trans. on Communications, Vol. 44, No. 10, pp. 1261-1271 (1996)
11. Benedetto S., Montorsi G.: Unveiling Turbo Codes: Some Results on Parallel Concatenated Coding Schemes. IEEE Trans. Inf. Theory, Vol. 42, No. 2, pp. 409-428, (1996)
12. Wiberg, N.: Codes and Decoding on General Graphs. Doctoral Dissertation, Department of Electrical Engineering, Linköping University, Sweden (1996)
13. Pearl, J.; "Fusion, Propagation and Structuring in Belief Networks", Artificial Intelligence, No. 29, pp. 241-268 (1986)
14. Richardson, T., Urbanke, R.: The Capacity of Low Density Parity Check Codes under Message Passing Decoding. IEEE Trans. Inf. Theory, Vol. 47, No. 2, pp. 599-618 (2001)
15. MacKay, D. J.: Good Error Correcting Codes based on Very Sparse Matrices. IEEE Trans. Inf. Theory, Vol. 45, pp. 399- 431 (1999)
16. Witten, I., Frank E.: Data Mining, Practical Machine Learning Tools and Techniques with JAVA Implementations. Morgan Kaufmann Publishers, San Francisco, California (2000)
17. Fossorier, M. P.C., Mihaljevic, M., Hideki, I.: Reduced Complexity Iterative Decoding of Low Density Parity Check Codes Based on Belief Propagation. IEEE Trans. on Comm., Vol. 47, pp. 673-680 (1999)

A Study on End-Cut Preference in Least Squares Regression Trees

Luis Torgo

LIACC-FEP, University of Porto
R.Campo Alegre, 823
4150 PORTO, PORTUGAL
email: ltorgo@liacc.up.pt,
URL: http://www.liacc.up.pt/∼ltorgo

Abstract. Regression trees are models developed to deal with multiple regression data analysis problems. These models fit constants to a set of axes-parallel partitions of the input space defined by the predictor variables. These partitions are described by a hierarchy of logical tests on the input variables of the problem. Several authors have remarked that the preference criteria used to select these tests have a clear preference for what is known as end-cut splits. These splits lead to branches with a few training cases, which is usually considered as counter-intuitive by the domain experts. In this paper we describe an empirical study of the effect of this end-cut preference on a large set of regression domains. The results of this study, carried out for the particular case of least squares regression trees, contradict the prior belief that these type of tests should be avoided. As a consequence of these results, we present a new method to handle these tests that we have empirically shown to have better predictive accuracy than the alternatives that are usually considered in tree-based models.

1 Introduction

Regression trees [6] handle multivariate regression methods obtaining models that have proven to be quite interpretable and with competitive predictive accuracy. Moreover, these models can be obtained with a computational efficiency that hardly has parallel in competitive approaches, turning these models into a good choice for a large variety of data mining problems where these features play a major role.

Regression trees are usually obtained using a least squares error criterion that guarantees certain mathematical simplifications [8, Sec. 3.2] that further enhance the computational efficiency of these models. This growth criterion assumes the use of averages in tree leaves, and can be seen as trying to find partitions that have minimal variance (i.e. squared error with respect to the average target value). The main drawback of this type of trees is the fact that the presence of a few outliers may distort both the average as well as having a strong influence in the choice of the best splits for the tree nodes. In effect, as we will see in this paper, the presence of outliers[1] may lead to the choice of split tests that have a very small sub-set of cases in one of the branches. Although these splits are the best according to the least squares error criterion they are counter-intuitive to the user and as we will see may even degrade predictive performance on unseen data. Users find it hard to understand that trees have top level nodes with branches that are very specific. Most users expect that top level nodes "discriminate" among the most relevant groups of observations (*e.g.* the observations with very high value of the target variable and the others).

The work presented in this paper addresses the problem of allowing this type of splits in regression trees, which is known as the end-cut preference problem [2, p.313-317]. We study this type of splits and their effect on both predictive accuracy and interpretability of the models. We compare this to the alternative of avoiding this type of splits in the line of what was proposed by Breiman and colleagues [2]. Our extensive experimental comparison over 63 different regression problems shows that the differences in terms of predictive accuracy of both alternatives are quite often statistically significant. However, the overall number of significant differences does not show a clear winner, which contradicts prior belief on the effect of end-cut preference in tree-based regression models.

In this paper we propose an alternative method that allows end-cut preference only in lower levels of the trees. The motivation behind this method is to avoid these splits in top level nodes, which is counter-intuitive for the users, but at the same time use them in lower levels as a means to avoid their negative impact in the accuracy of trees using least squares error criteria. Our experimental comparisons show a clear advantage of this method in terms of predictive accuracy when compared to the two alternatives mentioned before.

In the next section we present a brief description of least squares regression trees methodology and of the end-cut preference problem. Section 3 presents an experimental comparison between the alternatives of allowing and not allowing end-cut splits. In Section 4 we describe our proposed approach to handle the end-cut preference problem, and present the results of comparing it to the other alternatives. Finally, in Section 5 we provide a deeper discussion of the study carried out in this paper.

2 Least Squares Regression Trees

A regression tree can be seen as a kind of additive regression model [4] of the form,

$$rt\,(x) = \sum_{i=1}^{l} k_i \times I\,(x \in D_i) \tag{1}$$

where $k_i's$ are constants; $I\,(.)$ is an indicator function returning 1 if its argument is true and 0 otherwise; and $D_i's$ are disjoint partitions of the training data D such that $\bigcup_{i=1}^{l} D_i = D$ and $\bigcap_{i=1}^{l} = \phi$.

[1] These may be "real" outliers or noisy observations.

These models are sometimes called piecewise constant regression models. Regression trees are constructed using a recursive partitioning (RP) algorithm. This algorithm builds a tree by recursively splitting the training sample into smaller subsets. The algorithm has three key issues:

- A way to select a split test (the splitting rule).
- A rule to determine when a tree node is terminal.
- A rule for assigning a model to each terminal node (leaf nodes).

Assuming the minimization of the least squares error it can be easily proven (*e.g.* [8]) that if one wants to use constant models in the leaves of the trees, the constant to use in each terminal node should be the average target variable of the cases falling in each leaf. Thus the error in a tree node can be defined as,

$$Err\,(t) = \frac{1}{n_t} \sum_{D_t} (y_i - \overline{y}_t)^2 \tag{2}$$

where D_t is the set of n_t training samples falling in node t; and $\overline{y}_t$ is the average target variable (Y) value of these cases.

The error of a regression tree can be defined as,

$$Err\,(T) = \sum_{l \in \widetilde{T}} P\,(l) \times Err\,(l) = \sum_{l \in \widetilde{T}} \frac{n_l}{n} \times \frac{1}{n_l} \sum_{D_l} (y_i - \overline{y}_l)^2 = \frac{1}{n} \sum_{l \in \widetilde{T}} \sum_{D_l} (y_i - \overline{y}_l)^2 \tag{3}$$

where $\widetilde{T}$ is the set of leaves of tree T; and $P(l)$ is the probability of a case falling in leaf l (which is estimated with the proportion of training cases falling in the leaf).

During tree growth, a split test s, divides the cases in node t into a set of partitions. The decrease in error of the tree resulting from this split can be measured by,

$$\Delta Err\,(s,t) = Err\,(t) - \sum_i \frac{n_i}{n} \times Err\,(t_i) \tag{4}$$

where $Err\,(t_i)$ is the error on the subset of cases of branch i of the split test s.

The use of this formula to evaluate each candidate split would involve several passes through the training data with the consequent computational costs when handling problems with a large number of variables and training cases. This would be particularly serious, in the case of continuous variables that are known to be the major computational bottleneck of growing tree-based models [3]. Fortunately, the use of the least squares error criterion, and the use of averages in the leaves, allow for further simplifications of the formulas described above. In effect, as proven in [8], for the usual setup of binary trees where each node has only two sub-branches, t_L and t_R, the best split test for a node is the test s that maximizes the expression,

$$\frac{S_L^2}{n_{t_L}} + \frac{S_R^2}{n_{t_R}} \tag{5}$$

where $S_L = \sum_{D_{t_L}} y_i$ and $S_R = \sum_{D_{t_R}} y_i$.

This expression means that one can find the best split for a continuous variable with just a single pass through the data, not being necessary to calculate averages and sums of squared differences to these averages. One should stress that this expression could only be derived due to the use of the least squares error criterion and of the use of averages in the leaves of the trees[2].

Breiman and colleagues [2] mention that since the work by Morgan and Messenger [5] it is known that the use of the least squares criteria tends to favor end-cut splits, *i.e.* splits in which one of the branches has a proportion of cases near to zero[3].

To better illustrate this problem we describe an example of this end-cut preference occurring in one of the data sets we will use our experiments, the *Machine*[4] domain. In this data set the best split test according to the error criterion of Equation 5 for the root node of the tree, is the test $MMAX \leq 48000$. This split divides the 209 training cases in two sub-branches, one having only 4 observations. This is a clear example of a end-cut split. Figure 1 helps to understand why this is the best split according to the criterion of Equation 5.

As it can be seen in Figure 1, there are 4 observations (upper right part of the figure) that have end-cut values in the variable MMAX, and at the same time outlier values in the target variable. These are the two characteristics that when appearing together lead to end-cut splits. Within this context, a candidate split that "isolates" these cases in a single branch is extremely valuable in terms of the least squares error criterion of Equation 5.

Allowing splits like $MMAX \leq 48000$ in the example above, may lead to trees that seem quite ad-hoc to users that have a minimal understanding of the domain, because they tend to expect that top level nodes show highly general relations and not very specific features of the domain. This is reinforced by the fact that on most large data sets, trees do tend to be too deep for a user to grasp all details, meaning that most users will only be able to capture top-level splits. As such, although no extensive experimental comparisons have been carried out till now[5], it has been taken for granted that end-cut splits are undesirable, and most existing tree-based systems (*e.g.* CART [2], THAID [5] or C4.5 [7]) have some mechanism for avoiding them. However, if the drawbacks in terms of user expectations are irrefutable, as we will see in Section 3 the drawbacks of end-cut splits in terms of predictive accuracy are not so clear at all in the case of least squares regression trees.

[2] In [8] a similar expression was developed for the least absolute deviation criterion with medians on the leaves of the trees.

[3] For a formal proof of end-cut preference see [2, p.313-317].

[4] Available for instance in http://www.liacc.up.pt/∼ltorgo/Regression/DataSets.html.

[5] To the best of our knowledge.

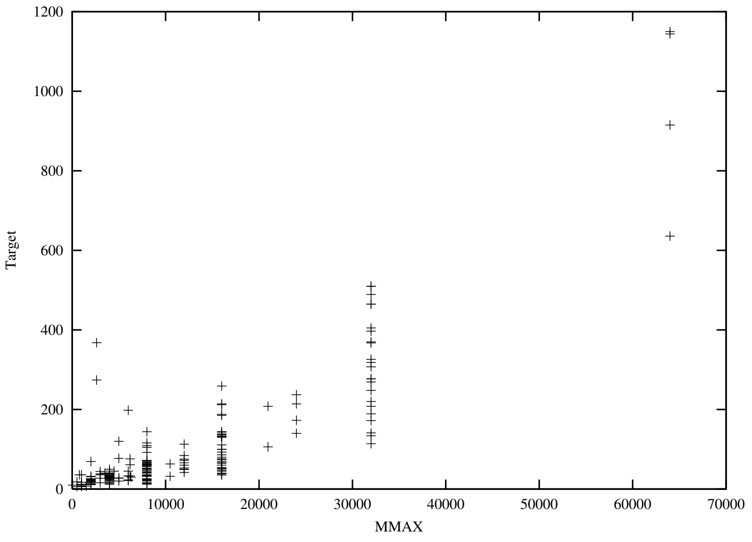


Fig. 1. An example of a end-cut preference problem.

3 An Experimental Analysis of the End-Cut Preference

In this section we carry out an experimental study of the consequences of end-cut splits. Namely, we compare the hypothesis of allowing this type of splits, and the alternative of using some form of control to avoid them.

In this experimental comparison we have used 63 different regression data sets. Their main characteristics (number of training cases, number of continuous variables, and number of nominal variables) are shown in Table 1.

Regarding the experimental methodology we have carried out 10 repetitions of 10-fold cross validation experiments, in the light of the recent findings by Bradford and Brodley [1] on the effect of instance-space partitions. Significance of observed differences were asserted through paired t-tests with 95 and 99 confidence levels.

The first set of experiments we report compares the following two types of least squares regression trees[6]. The first tree has no control over end-cut splits, thus allowing them at any stage of the tree growth procedure as long as they are better according to the criterion of Equation 5. The second type of trees does not allow splits[7] that lead to branches that have less cases then a minimum value

[6] Both are implemented in system RT (http://www.liacc.up.pt/~ltorgo/RT/), and they only differ in the way they handle end-cut splits. All other features are the same.

[7] Both on continuous as well as nominal variables.

Table 1. The Used Data Sets.

Data Set	*Characteristics*	*Data Set*	*Characteristics*
Abalone (Ab)	4177; 7; 1	Elevators (El)	8752; 40; 0
Delta Elevators (DE)	9517; 6; 0	Ailerons(Ai)	7154; 40; 0
Kinematics (Ki)	8192; 8; 0	Telecomm (Te)	15000; 26; 0
ComputerA (CA)	8192; 22; 0	ComputerS (CS)	8192; 12; 0
Algal1 (A1)	200; 8; 3	Algal2 (A2)	200; 8; 3
Algal3 (A3)	200; 8; 3	Algal4 (A4)	200; 8; 3
Algal5 (A5)	200; 8; 3	Algal6 (A6)	200; 8; 3
Algal7 (A7)	200; 8; 3	Anastesia (An)	80; 6; 0
Auto-Mpg (AM)	398; 4; 3	Auto-Price (AP)	159; 14; 1
Bank8FM (B8)	4500; 8; 0	Bank32NH (B32)	4500; 32; 0
CloseNikkei (CN)	2000; 49; 1	CloseDow (CD)	2399; 49; 1
Chlorophyll (Chl)	72;4;1	House8L (H8)	22784; 8; 0
House 16H (H16)	22784; 16; 0	Diabetes (Di)	43; 2; 0
Pyrimidines (Py)	74; 24; 0	Triazines	186; 60; 0
FacultyD5001 (Fa)	197; 33; 0	Employment (Em)	368; 18; 0
ArtificialD2 (D2)	40768; 2; 0	Industry (In)	1555; 15; 0
Friedman Example (Fr)	40768; 10; 0	Housing (Ho)	506; 13; 0
Machine CPU (Ma)	209; 6; 0	Marketing (Mkt)	944; 1; 3
Artificial MV (MV)	25000; 7; 3	Puma8NH (P8)	4500; 8; 0
Puma32NM (P32)	4500; 32; 0	Servo	167; 0; 4
WiscoinBreastCancer (WBC)	194; 32; 0	CaliforniaHousing (CH)	20460; 8; 0
Additive (Ad)	30000; 10; 0	1KM (1KM)	710; 14; 3
Acceleration (Ac)	1732; 11; 3	CO2-emission (CO2)	1558; 19; 8
CW Drag (CW)	1449; 12; 2	Available Power (AP)	1802; 7; 8
Driving Noise (DN)	795; 22; 12	Fuel Town (FTw)	1764; 25; 12
Fuel Total (FTo)	1766; 25; 12	Fuel Country (FC)	1764; 25; 12
Maximal Torque (MT)	1802; 19; 13	Top Speed (TS)	1799; 17; 7
Maintenance Interval (MI)	1724; 6; 7	Heat (He)	7400; 8; 4
Steering Acceleration (SAc)	63500; 22; 1	Steering Angle (SAn)	63500; 22; 1
Steering Velocity (SV)	63500; 22; 1	Fluid Discharge (FD)	530; 26; 6
Fluid Swirl (FS)	530; 26; 6	China (Ch)	217; 9; 0
Delta Ailerons (DA)	7129; 5; 0		

established by the user. In our experiments we have set this minimum value to 10 cases.

Table 2 shows the results of the comparison between the two alternatives in terms of Normalized Mean Squared Error (NMSE). Columns two and three show the data sets where we observed a statistically significant win of some method at different confidence levels, and the fourth column shows the cases where the observed differences were not statistically significant.

Table 2. End-Cut versus No End-Cut in terms of NMSE.

	99%	95%	Not significant
End-Cut Wins	**20** Ad,AP,B32,Chl,CO2,CW,D2, FS,FTo,FTw,He,Ma,MI,MT, An,Se,SAc,SAn,SV,TS	**2** FC,FD	**15** Ac,A1,A2,A3,A5,A6,A7, Ch,CN,Fa,DN,Ho,Te,AP,WBC
No End-Cut Wins	**17** 1KM,Ab,Ai,B8,CH,CD,CA, CS,DA,DE,El,Fr,H16,H8,Ki P32,P8	**2** In,Py	**7** A4,AP,Di,Mkt,MV,Em,Tr

The first thing to remark is that there is a statistically significant difference between the two approaches on 41 of the 63 data sets. This reinforces the importance of the question of how to handle end-cut splits. However, contrary to our prior expectations based on previous works (*e.g* [2]), we didn't observe a clear advantage of not using end-cut splits[8]. On the contrary, there is a slight advantage of the alternative allowing the use of end-cut splits at any stage of tree growth (the average NMSE over all data sets of this alternative is 0.4080, while not allowing end-cut splits leads to an average NMSE of 0.4140). The main conclusion to draw from these results is that they provide strong empirical evidence towards the need of a re-evaluation of the position regarding end-cut splits in the context of least squares regression trees.

Why should end-cut splits be beneficial in terms of predictive error? We believe the best answer to this question is related to the statistic used to measure the error. Least squares regression trees revolve around the use of averages and squared differences to these averages (*c.f.* Section 2). The use of averages as a statistic of centrality for a set of cases is known to suffer from the presence of outliers. By not allowing the use of end-cut splits that tend to isolate these outliers in a separate branch (*c.f.* Figure 1), every node will "suffer" the influence of these extreme values (if they exist). This will distort the averages, which may easily lead to larger errors as the predictions of the trees are obtained using the averages in the leaves. Going back to the example described in Section 2 with the *Machine* data set, if one does not allow the use of end-cut splits, instead of

[8] Still, we must say that the method proposed in [2] for controlling these splits is slightly different from the one use d in our experiments.

immediately isolating the four outlier cases shown in Figure 1, they will end-up falling in a leaf that includes 10 other observations. This leaf has an average target variable value of 553.64, which will be the prediction of the tree for every test case falling in this leaf. However, 10 out of the 14 observations in this leaf, have a target value in the range [208..510]. Thus the distribution in this leaf is clearly being skewed by the outliers and this provides an idea of the risk of using this leaf to make predictions. The same conclusion can be reached by looking at the Mean Squared Errors[9] at the leaves of both trees. While the tree using end-cut splits has an average MSE over all leaves of 5132.2, the tree without end-cut splits has and average of 15206.4, again highlighting the effect of these outliers, that clearly increase the variance in the nodes. One should remark that this does not mean that the tree using end-cut splits is overfitting the data, as both trees went through the same post-pruning process that is supposed to eliminate this risk (moreover the experimental comparisons that were carried out show that this is not occurring, at least in a consistent way).

In resume, although clearly going against the intuition of users towards the generality of the tests in the trees, end-cut splits provide some accuracy gains in several data sets. This means that simply eliminating them can be dangerous if one is using least squares error criteria. We should stress that the same conclusions may not be valid if other error criteria were to be used such as least absolute deviations, or even the criteria used in classification trees, as these criteria do not suffer such effects of outliers.

As a consequence of the experimental results reported in Table 2 we propose a new form of dealing with end-cut splits that tries to fulfill the interpretability expectations of users that go against the use of end-cut splits, while not ignoring the advantages of these splits in terms of predictive accuracy. This new method is described in the next section.

4 A Compromising Proposal for Handling End-Cut Preference

The main idea behind the method we propose to deal with end-cut preference is the following. End-cut splits should not be allowed in top level nodes of the trees as they handle very specific (poorly represented in the training sample) areas of the regression input space, thus going against the interpretability requirements of most users. As such, our method will use mechanisms to avoid these splits in top level nodes of the trees, while allowing them in bottom nodes as a means to avoid the distorting effects that outliers have in the averages in the leaves.

In order to achieve these goals we propose a simple method consisting of not allowing end-cut splits unless the number of cases in the node drops below a certain user-definable threshold[10]. Moreover, as in the experiments of Section 3, a test is considered an end-cut split if one of its resulting branches has less than

[9] Larger values of MSE indicate that the values are more spread around the average.
[10] In the experiments we will report we have set this threshold to 100 cases.

a certain number of cases[11]. This means that nodes with a number of training cases between the first and second of these thresholds are allowed to consider end-cut splits. These are the bottom nodes of the trees[12].

Our hypothesis is that with this simple method we will obtain trees that are acceptable in terms of interpretability from the user perspective, but at the same time will outperform both alternatives considered in Section 3 in terms of predictive accuracy. With the purpose of testing this hypothesis we have carried out an experiment similar to the one reported in Section 3, but now comparing our proposed method with the two alternatives of allowing end-cut splits everywhere, and not allowing them at all. The results of comparing our method to the former alternative are shown in Table 3.

Table 3. Our method compared to allowing end-cut splits in terms of NMSE.

	99%	95%	Not significant
Our Method Wins	**16** Ab,Ad,Ai,CH,CD,CA,D2,DA, DE,El,H16,H8,In,Ki,Pu8,SV	**0**	**22** 1KM,A2,A3,A5,AP,B8,CS DN,FD,FTo,FTw,FC,He,MI Mkt,Em,Pu32,SAc,SAn,TS,Tr
End-Cut Wins	**2** A1,Fa	**0**	**23** Ac,A6,A7,AM,B32,Chl,Ch, CN,CO2,CW,Di,FS,Fr,Ho,Ma, MT,MV,Te,AP,Py,An,WBC,Se

This comparison clearly shows that there is no particular advantage in allowing end-cut splits everywhere, when compared to our proposal (with two single exceptions). Moreover, our proposal ensures that this type of splits will not appear in top level nodes of the trees[13], which fulfills the user's expectations in terms of interpretability of the models. In effect, going back to the *Machine* example, with our proposal we would not have a root node isolating the four outliers (as with the alternative of allowing end-cut splits), but they would still be isolated in lower levels of the tree[14].

What this comparison also shows is that our proposal can outperform the method of allowing end-cut splits in several data sets. It is interesting to observe that most of these 16 cases are included in the set of 17 significant losses of the alternative allowing end-cut splits shown in Table 2.

[11] We have used the value of 10 for this threshold.

[12] Unless the training sample is less than 100 cases, which is not the case in all but four of our 63 benchmark data sets (*c.f.* Table 1).

[13] Unless the data set is very small.

[14] Namely, the root node would consist of the split $MMAX \leq 28000$, which divides the 209 training cases in 182 and 27, respectively, and then the 27 cases (that include the 4 outliers) would be split with the end-cut test $MMAX \leq 48000$ (*c.f.* Figure 1 to understand what is being done with these splits).

The results of comparing our method to the alternative of not allowing end-cut splits are shown in Table 4.

Table 4. Our method compared to not allowing end-cut splits in terms of NMSE.

	99%	95%	Not significant
Our Method Wins	**22** Ad,AP,B32,Chl,CD,CO2,CW,D2, FD,FS,FTo,FTw,He,Ma,MI,MT, An,Se,SAc,SAn,SV,TS	**2** FC,H8	**13** Ac,A2,A3,A5,A6,A7, DE,DN,H16,In,Mkt,AP,Tr
No End-Cut Wins	**10** 1KM,Ai,B8,CH,CA, CS,El,Fr,Ki,Pu32	**4** A1,Fa,Pu8,Py	**12** Ab,A4,AM,Ch,CN,DA, Di,Ho,MV,Em,Te,WBC

Once again we observe a clear advantage of our proposal (24 significant wins), although there are still 14 data sets where not allowing end-cut splits seems to be preferable. However, comparing to the alternative of always allowing end-cut splits, which has clear disadvantages from the user interpretability perspective, our method clearly recovers some of the significant losses (*c.f.* Table 2). It is also interesting to remark that with the single exception of the *H8* data set, all 22 wins of the strategy using end-cut splits over the no-end-cuts approach, are included in the 24 wins of our proposal. This means that our method fulfills our objective of being able to take advantage of the gains in accuracy entailed by the use of end-cut splits, in spite of not using them in top levels of the trees.

In spite of the advantages of our proposal, there are also some drawbacks that should be considered. Namely, there is a tendency for producing larger trees (in terms of number of leaves) than with the other two alternatives that were considered in this study. This is reflected in the results shown in Table 5, that presents the comparison in terms of number of leaves of our proposal with the other two alternatives.

Table 5. Tree size comparison of our method with the other two alternatives.

	No End-Cut Splits			All End-Cut Splits		
	99%	95%	Not significant	99%	95%	Not significant
Our Wins	**23**	**2**	**3**	**1**	**0**	**15**
Our Losses	**31**	**0**	**5**	**20**	**2**	**25**

These results seem to contradict our goal of a method that produces trees more interpretable to the user than the trees obtained when allowing end-cut splits. Interpretability is known to be a quite subjective issue. Still, we claim that in spite of having a larger number of leaves (*c.f.* Table 5), the trees obtained with our method are more comprehensible. As we have mentioned before, in most real-world large data sets, the trees obtained by this type of systems are too large for any user to be able to grasp all details. As such, we argue that only top-level nodes are in effect “understood” by the user. As our method does not allow end-cut splits in these top level nodes, we claim that this leads to trees that are more comprehensible to the user.

5 Discussion

The results of our empirical study on the effects of end-cut preference within least squares regression trees, lead us to the conclusion that there is no clear winner among the two standard alternatives of allowing or not allowing the use of end-cut splits. These results are somehow surprising given the usual position regarding the use of these splits. However, our study confirms the impact of the method used to handle these splits on the predictive accuracy of least squares regression trees. Our analysis of the reasons for the observed results indicates that this study should not be generalized over other types of trees (namely classification trees).

The method we have proposed to handle end-cut splits is based on the analysis of the requirements of users in terms of interpretability, and also on the results of our empirical study. By allowing end-cut splits only in lower levels of the trees, we have shown that it is possible to outperform the other two alternatives considered in the study in terms of predictive accuracy. Moreover, this method avoids end-cut splits in top level nodes which goes in favor of user expectations in terms of comprehensibility of the trees. However, our results also show that there is still some space for improvements in terms of predictive accuracy when compared to the alternative of not allowing end-cut splits. Future work, should be concentrated in trying to find not so ad-hoc methods of controlling these splits so as to avoid some of the still existing significant losses in terms of predictive accuracy. Moreover, the bad results in terms of tree size should also be considered for future improvements of our proposal.

6 Conclusions

We have described an empirical study of the effect of end-cut preference in the context of least squares regression trees. End-cut splits have always been seen as something to avoid in tree-based models. The main conclusion of our experimental study is that this assumption should be reconsidered if one wants to maximize the predictive accuracy of least squares regression trees. Our results show that allowing end-cut splits leads to statistically significant gains in predictive accuracy on 22 out of our 63 benchmark data sets. In spite of the disadvantages of end-cut splits in terms of the interpretability of the trees from the user perspective, these experimental results should not be disregarded.

We have described a new form of dealing with end-cut splits that tries to take into account our empirical observations. The simple method we have described shows clear and statistically significant improvements in terms of predictive accuracy. Still, we have also observed that there is space for further improvements.

Future work should try to improve the method we have described, and also to carry out similar studies for tree-based models using different error criteria.

References

1. J. Bradford and C. Broadley. The effect of instance-space partition on significance. *Machine Learning*, 42(3):269–286, 2001.
2. L. Breiman, J. Friedman, R. Olshen, and C. Stone. *Classification and Regression Trees*. Statistics/Probability Series. Wadsworth & Brooks/Cole Advanced Books & Software, 1984.
3. J. Catlett. *Megainduction: machine learning on very large databases*. PhD thesis, Basser Department of Computer Science, University of Sidney, 1991.
4. T. Hastie and R. Tibshirani. *Generalized Additive Models*. Chapman & Hall, 1990.
5. J. Morgan and R. Messenger. Thaid: a sequential search program forthe analysis of nominal scale dependent variables. Technical report, Ann Arbor: Institute for Social Research, University of Michigan, 1973.
6. J. Morgan and J. Sonquist. Problems in the analysis of survey data, and a proposal. *Journal of American Statistics Society*, 58:415–434, 1963.
7. J. Quinlan. *C4.5: programs for machine learning*. Kluwer Academic Publishers, 1993.
8. L. Torgo. *Inductive Learning of Tree-based Regression Models*. PhD thesis, Faculty of Sciences, University of Porto, 1999.

The Use of Domain Knowledge in Feature Construction for Financial Time Series Prediction

Pedro de Almeida[1] and Luís Torgo[2]

[1] Physics Department, Universidade da Beira Interior, 6200 Covilhã, Portugal
`nop00997@mail.telepac.pt`
[2] LIACC, Rua Campo Alegre, 823 – 2º, 4150 Porto, Portugal
`ltorgo@liacc.up.pt`

Abstract. Most of the existing data mining approaches to time series prediction use as training data an embed of the most recent values of the time series, following the traditional linear auto-regressive methodologies. However, in many time series prediction tasks the alternative approach that uses derivative features constructed from the raw data with the help of domain theories can produce significant prediction accuracy improvements. This is particularly noticeable when the available data includes multivariate information although the aim is still the prediction of one particular time series. This latter situation occurs frequently in financial time series prediction. This paper presents a method of feature construction based on domain knowledge that uses multivariate time series information. We show that this method improves the accuracy of next-day stock quotes prediction when compared with the traditional embed of historical values extracted from the original data.

1 Introduction

Recently, several data mining techniques have been applied with success to time series prediction based on samples of historical data (*e.g.* [1], [5], [20]). Most of these approaches use supervised machine learning techniques and a data preparation stage that produces a set of examples in a two-dimension, tabular, “standard form” [28]. Several approaches can be used to prepare this kind of tabular representation from the time series raw data. Choosing the most appropriate set of features for the time series problem in hand can have a significant impact on the overall accuracy of the prediction models. This is the main problem addressed in this paper, for the particular case of financial time series prediction.

1.1 Traditional Temporal Embed

The simplest and most common procedure of transforming the original time series data to produce a set of examples in tabular form is to use the last known values of the time series as features describing each example, and using the next value of the time series as the respective target value of the example. This auto-regressive data preparation technique is usually called "time-delay embedding" [22], "tapped delay line" or "delay space embedding" [17].

To illustrate the basic temporal embedding, let us consider an univariate time series

$$X(t) = \ldots, x_{t-2}, x_{t-1}, x_t, x_{t+1}, x_{t+2}, \ldots$$

from which the values up to x_t (that is, …, x_{t-2}, x_{t-1}, x_t) are known. Let us further consider as our objective the construction of a model to predict the next value of the same time series (x_{t+1}), using some supervised data mining process. Assuming that this depends at most on the k previous values, traditional time embed approaches will describe each example using k features, each taking one of the k previous measurements of the time series (x_t, x_{t-1}, x_{t-2}, …, $x_{t-(k-1)}$). Each complete example will thus have the form

$$x_t, x_{t-1}, x_{t-2}, \ldots, x_{t-(k-1)}, x_{t+1}$$

where x_{t+1} is the value of the target (dependent) variable.

This kind of data preparation forms the basis of the classical autoregressive time series prediction methods like AR [29] or ARMA [6], and is theoretically justified by the Takens theorem [24]. This theorem states that, with some restrictions, a number of $(2 \cdot N)+1$ past values is enough to reconstruct the model of a noiseless system with N dimensions. In the case of the described $X(t)$ time series, assuming absence of noise, and considering the series to be generated by a N-dimensional system, the features x_t, x_{t-1}, x_{t-2}, …, $x_{t-(2N)}$ would be enough for the extraction of the system model and for predicting future values of the time series.

1.2 Limitations of the Temporal Embed

It should be remarked that many real-life systems suffer complex interactions with other systems and, even if they have an intrinsic linear behavior, they tend to generate time series that present severe problems to comply with the Takens theorem restrictions. In particular, this theorem does not seem to apply to most financial time series, namely, to stock exchange time series.

In effect, stock exchange time series are generated by extremely complex systems that involve the interaction of thousands or millions of independent agents (the investors) each capable of changing its behavior over short time frames according to a virtually limitless number of possible individual "states", resulting in a system with a number of dimensions that, in practical terms, must be considered infinite [14]. This fact implies that the number of historical values needed to construct each example in a way conforming to the Takens theorem conditions would be unrealistic (the necessary data would not be available and would be unmanageable by the machine learning algorithms). Moreover, the system global behavior is not static over time (for instance, the number and individual characteristics of the intervening agents are not constant). This non-stationary behavior of the system means that "old" time series data may not be truly representative of the current system (in fact, it was generated by a related but different system that no longer exists) and using them blindly to generate predictions for the current system can be misleading.

These characteristics of stock exchange time series result in the almost unanimous admittance that base data consisting solely of historical values of the time series being predicted do not contain enough information to explain more than a fraction of the variance in the time series. In fact, the Efficient Markets Theory [8], now somewhat discredited ([1], [11], [15]) but still accepted as basically correct by many economists, states that those data do not contain any information useful for the prediction of the future behavior of the system.

The problems that impair the applicability of Takens Theorem to very complex dynamic systems do not seem to be reduced by the use of multivariate information as basis data for time series modeling. As an example, the original data we use in our experiments includes 7 values for each day and each stock. This corresponds to 7 time series that are available to be used as basic data for model construction. In this context, using all the 7 time series with an embed dimension of, say, 25 (obviously insufficient to include enough information to describe the system), would result in 175 features describing each example. Even if those features were discretized to adopt a maximum of 5 different values, that would create a "representation space" with the capacity of distinguishing 175^5 different cases. This kind of input dimension is obviously too large considering the number of training and testing examples available (we have 10 years of daily data corresponding to around 2400 records), and would result in a very sparsely populated space that would severely limit the efficiency of most machine learning algorithms [3], [23]. This problem is reinforced by the fact that, due to the system complexity and the lack of sufficient information on the base data, the system behaves as if each variable (dependent and independent) includes a strong noise component. Thus, since all the information present in the base data only explains a small proportion of the system variance and the information present in our 175 variables will explain an even smaller proportion of that variance, we can expect each of those variables to have a very small individual relevance to the desired prediction.

1.3 Alternatives to the Temporal Embed

In situations where heavy overfitting problems can be expected due to the sparsely populated representation space and when noisy data are associated with a possibly large number of input variables with low individual value to knowledge extraction, it is frequently possible to obtain better results by combining several of the original variables [16], [19]. The aim of developing such feature combinations is the creation of a reduced set of "derivative" variables having a greater discriminative power for the modeling task being considered.

One of the possible approaches to the development of a reduced set of derivative variables that contain most of the useful information present in the original data is the use of an automated method that searches for some combination of the original variables. The most used of such methods is "Principal Component Analysis" [4]. This method develops orthogonal linear combinations of the original variables and ranks them on the basis of their ability to "explain" the target variable variance. The main problem with this approach is that the original variables are replaced by the most significant linear combinations and so, the data mining algorithms will no longer be able to search for non-linear combinations of the original variables. This approach is therefore particularly appropriate for "data reduction" when only linear prediction methods will be used, and on problems related to systems known at start to be basically linear (clearly not the case of stock exchange time series [14], [31]).

Another approach to feature construction consists in the "manual" development of a reduced set of derivative variables using domain knowledge. This approach is particularly efficient in domains where there is an available (but incomplete, thus not allowing a deterministic prediction) body of domain theory relating the data to the system mechanics and behavior. In stock exchange prediction there is no sure domain knowledge and, considering only historical stock exchange information as base data, the available domain theories (related to "technical analysis" of stocks [7], [18]) are particularly uncertain [10]. However, the large number of known technical analysis indicators[1] seems a good starting point to build derivative features. In effect, the highly uncertain applicability of these indicators to individual stocks, markets, and time frames, limits them as final global theories, but does not prevent their usefulness to construct input features for machine learning algorithms, since most of these algorithms can filter out the features that prove to be least useful.

2 Data Pre-processing in Financial Time Series Prediction

A review of the published works on financial time series prediction shows that, in spite of the limitations mentioned in Section 1.2, most works use either the basic version of temporal embed or very limited transformations of this technique.

As examples of the use of direct temporal embed, we can mention the use of a direct univariate embed of dimension two to predict particularly strong daily stock exchange quotes variations [20], or the use of a multivariate basic embed of daily stock quotes information as input data to a genetic algorithm used to conduct a direct search for trading criteria in [30].

Regarding the use of basic transformations of temporal embed we can refer the approach followed by [13], who employs the difference of the logarithms of the last values of several time series to predict the stock quotes of several Japanese companies, or the work of [15] where logarithmic transformations of the differences of the last two and three known values of exchange rate times series are used to predict the variation direction of those time series.

Interesting examples of more ambitious adaptations of the basic temporal embed can be found in two entries of the important time series prediction competition carried out in Santa Fe in 1992 [25]: Mozer tests several transformations involving different weights for the embedded values, in an univariate context related to the prediction of exchange rates between the US dollar and the Swiss franc [17] and, for the prediction of the same time series, Zang and Hutchinson try a univariate embed using non-consecutive past values for different prediction time frames [31].

Although not so frequent, it is also possible to find the use of more sophisticated variables derived from the base data. An early effort is the work described in [26],

[1] Descriptions can be found in a variety of sources, like, for instance, http://traders.com, or http://www.tradersworld.com)

where a neural network with 61 inputs is used for the prediction of the next day value of the US dollar / German mark exchange rate. Forty five of the 61 variables used in this work correspond to an embed of the times series being modeled, while the other 16 variables are developed from multivariate data that the authors believe to be relevant for the prediction of the time series. The work presented in [27] is another example involving more sophisticated derivative variables and more complete multivariate base data. In this latter study, the objective is also the prediction of the US dollar / German mark exchange rate. The authors use 69 input variables for the neural network that produces the predictions, but none of those variables are the result of a direct embed of the time series being predicted. In fact, 12 of the 69 variables are "built" based on past values of the time series, using typical "technical analysis" transformations, and the other 57 variables reflect multivariate fundamental information exterior to the time series being predicted (using data associated to exchange rates between other currencies, interest rates, etc.). In these two works, the large number of variables "fed" to the machine learning algorithms will lead to overfitting problems and, in effect, both works include as main objectives the presentation of new techniques to reduce overfitting problems in neural networks.

3 A System for Stock Quotes Prediction

When the goal is the short-term (up to a week) prediction of stock quotes, the relevance of fundamental information (both micro or macro-economic) tends to be small. In effect, even if this kind of information proves useful to predict a part of the long-term variability of the stocks, the proportion of that ability with direct reflection on the variance of the next few days would be very small [11], [18]. Thus, it seems reasonable to believe that most of the short term variability of stock values is due to fluctuations related to the behavior of the investors, which technical analysis claims that can be inferred from past variations of the stock quotes and volumes. However, there are still important problems to address, if an approach based on derived variables built with the help of domain knowledge is to be tried. These problems are related to the uncertain nature of the existing technical analysis "indicators", and to the vast number of existing indicators (the direct use of a large number of derivative variables of this kind could lead to overfitting problems similar to those related to the use of a large direct embed [10]).

An approach based on derived variables would benefit from a practical way of representing domain knowledge (in the form of technical analysis indicators) and also from an efficient way of generating variables from that representation. Also, given the large quantity of possible derived "technical variables" it may be advantageous to perform some kind of automatic filtering of the less relevant features. To try to circumvent the problems associated with this approach, we have developed an integrated prediction system that includes a knowledge representation language that allows the direct description of most technical analysis indicators using pre-defined language elements. Based on these descriptions the system is able to automatically generate from the raw data the features described by the user. This general set-up could be used with any machine learning algorithm that copes well with noisy data.

We chose to test this framework with a *k*-nearest neighbor algorithm [5], and with a regression version of the PA3 rule induction algorithm [2].

The *k*-NN algorithm uses a distance metric that involves a ranking of feature importance, while the rule induction algorithm does not need such ranking. However, due to the domain characteristics mentioned before, this system also gains from a reduction of the number of features. As such, our prediction system includes a feature ranking and selection step. The complete system architecture is shown in Figure 1.

The domain knowledge representation language allows a domain expert to define technical analysis indicators using a simple but flexible representation. As examples, consider the following instructions of this language:

percent(clo(i),clo(i+1)); i=0..3
ratio(moa(vol,3,0),moa(vol,15,0))

The first of these instructions describes 4 features, each corresponding to the percent variation between the last four daily closing values of a stock and the respective previous closing value. The second instruction describes a single feature corresponding to the ratio between the moving average of the last 3 daily trading volumes of the stock and the moving average of the last 15 daily trading volumes[2].

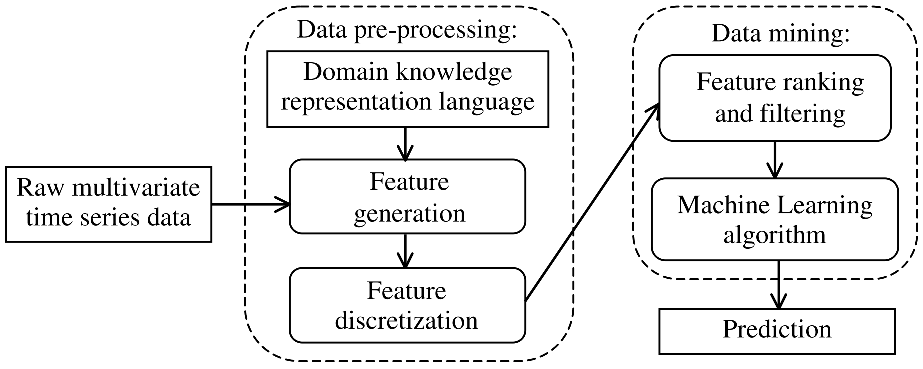

Data pre-processing:
Domain knowledge representation language
Raw multivariate time series data
Feature generation
Feature discretization
Data mining:
Feature ranking and filtering
Machine Learning algorithm
Prediction

Fig. 1. The complete time-series prediction system

The features defined with the language are automatically generated from the raw data and appended with the respective target value for each generated example, to form a set of examples in the tabular "normal form".

The resulting features can have very different discrete or continuous values. Discrete features can have ordered integer values (for instance, consider a feature that counts the number of times the closing value is higher than the opening value during the last 20 trading days), but can also have non-ordered values (e.g. a feature that represents the weekday of the last trading session). Continuous features can also have very different ranges of values. Given these different characteristics a feature

[2] The third parameter in the *moa* constructor specifies the number of days between the last value used in the moving average and the present reference example. This way, the expression *ratio(moa(vol,3,0),moa(vol,3,1))* relates two 3-day moving averages of the volumes (the first using the last 3 values and the second using the first 3 of the last 4 values).

discretization module is run before the set of examples is “fed” to the data mining algorithms. This module discretizes each feature into 5 values using a traditional approach [9] that is known to behave well over noisy data. This discretization technique works individually over each feature. It starts by analyzing the examples of the training set and then divides the range of values of each feature into five intervals with the same number of values in each interval. Each of these intervals is represented by an integer value ranging from 1 to 5, and the examples are discretized into one of these 5 discrete values accordingly. This simple approach to feature discretization produced robust results over our data. As some algorithms are able to work with non-discretized data, we have checked whether by using this discretization step some information was being lost. We have not noticed any significant accuracy differences in these tests and thus we have decided to work only with the discretized data.

After generating the training examples, our prediction system starts the core data mining step with a feature ranking and selection module. The feature selection procedure aims to reduce the overfitting problems related to the difficult domain characteristics mentioned in Section 1.2. In effect, some algorithms that use representation languages with larger descriptive power, are able to fit almost perfectly the hypersurface defined by the training data. This is the case of our rule induction PA algorithm. These methods would have a tendency to find existing but non-meaningful statistic fluctuations on the training data if given enough irrelevant features. To prevent this kind of overfitting, we include a feature ranking and filtering step in our system. This step uses a combination of the following feature relevance measures: the information gain; the Pearson’s *r* correlation between each feature and the target values; and a powerful feature relevance metric that evaluates the features in the context of the other features [12]. The first two of these metrics look to each feature individually, while the third considers feature interactions. These three measures are combined to form an overall ranking, using a direct averaging of the three independent feature rankings. Regarding the selection of features we simply retain the highest ranking features up to a certain number[3].

The *k*-NN algorithm we have used in our prediction system is inspired in Bontempi’s work [5]. It uses an orthogonal “Manhattan” distance metric that weights differently the features and a linear kernel function that gives greater weights to the training examples found to be nearest to the test example being predicted. After several tests over stock exchange time series[4], we opted to use a fixed neighborhood of 150 cases. This considerable number of neighbors seems to result in a good balance between bias and variance and has an important effect in the reduction of overfitting problems typically associated to the very noisy examples sets expected in stock exchange time series prediction.

The rule induction algorithm that was also used in our prediction system is a regression variant of a general-propose sequential-cover algorithm called PA3, which handles two-class problems. This algorithm was developed to be efficient over noisy data, and performed well when compared with other rule induction algorithms over several noisy data sets (including stock exchange time series prediction) [2]. PA3

[3] In this work we have used 10 features in all tests.

[4] This parameter tuning was carried out using only the sub-set of stock data that was used for training purposes according to the partitioning that will be described in Section 4.

induces an ordered list of “if…then…” rules. Each rule has the form “if <complex> then predict <class>”, where <complex> is a conjunct of feature tests, the “selectors”. In PA3, each selector implies testing a feature to see if its value is included in a specified range of values. The postcondition of a PA3 rule is a single Boolean value that specifies the class that the rule predicts. To produce a regression (numeric) prediction instead of a class prediction, we have used the very simple approach of assigning as postcondition of each rule the mean value of the training examples covered by the rule. This variant of the original PA3 algorithm was also adapted to accept (and to use in the rule evaluation) numeric values instead of class labels in the training examples.

Those two algorithms where chosen to be integrated in our prediction system not only because they tend to be efficient over noisy data, but also because they use totally different algorithmic approaches. This is an important issue since we wanted to test the validity of our different data pre-processing methods. In effect, it is conceivable that some data preparation approaches lead to examples sets better suited for specific machine learning algorithms, but less efficient when used by others.

4 Experimental Testing

The main goal of this paper is to show that the use of domain knowledge to generate derivative features from the raw data leads to better predictive accuracy results than the traditional embed approach, within financial time series prediction problems. To test this hypothesis, we have carried out a set of comparative experiments. In our experiments we have used time series data concerning five of the more actively traded companies listed in the Portuguese BVLP stock exchange: “BCP”, “Brisa”, “Cimpor”, “EDP” and “PT”. The base data consists of 855 daily records for each of the five companies (from 25 November 1997 to 11 May 2001). Each of those daily records includes 7 base variables: the date of the day; the closing value for the BVLP30 stock index; the volume of traded stocks; and the opening, maximum, minimum and closing values of the stock.

Regarding the methodology used to compare the different alternatives considered in this work we have used the following strategy. The available data for each company was split in four sections. The first 50 records were kept aside for the construction of the first processed example (thus allowing the use of embed dimensions of up to 50). The next 400 records were used to generate 400 training examples according to the three strategies that will be compared in this paper. The following 400 records were used to construct 400 testing examples. The last 5 records where kept aside to allow the use of target values up to five days ahead in the examples.

With respect to the method used to obtain predictions for the 400 test cases we have used the following strategy. Initially all learning algorithms were given access to the first 400 examples. With this set of examples a prediction is made for the first test case. After this prediction is obtained, this test example is added to the set of training cases leading to a new training set, now with 401 examples, which is used to obtain a new model. This iterative train+test process (sometimes known as sliding window) continues until a prediction is obtained for all 400 test examples. This means that for instance the prediction for the last test example is obtained with a model that was learned using 799 training cases (the original 400 training examples, plus the first 399 test examples).

Given the goal of our comparative experiments, we have used a prediction horizon of one single day. Regarding the target variable (the times series to predict) we have chosen an average of the next day quotes of each stock. This average consists of the mean of the opening, maximum, minimum and closing values of the stock during the following day. We called this average the "reference value" of the stock. More precisely, we predict the percentage change between the last known reference value and the next day reference value,

$$y(t) = 100 \times \frac{Ref(t) - Ref(t-1)}{Ref(t-1)} \tag{1}$$

where $Ref(t) = \frac{Close(t) + Max(t) + Min(t) + Open(t)}{4}$.

The reason for the use of this "reference value" instead of using, for example, the closing value of each day, is related to the nearly stochastic nature of these time series. In effect, this kind of time series behaves as if it had an important component of added white noise. Thus, every single observation of the time series is affected by a random amount of noise, which tends to "drown" the relatively small variations of the stock values attributable to the system "real" behavior ([8], [10]). The use of this reference value results in two main advantages when compared with the use of a single daily point value of the stock quotes (for instance the closing value). The first, and most important, is the reduction in the proportion of the variance that results from random noise with relation to the variance resulting from the subjacent system behavior. The second advantage is related to the usefulness of the predicted values for trading. To illustrate this latter advantage, let us suppose that we predict a 1% rise in the reference value for the next day. Based on that prediction, we could place a sell order with a (minimum) sell value equal to the predicted reference value. In this situation, we can attain the sell value during the next day even if the predicted rise for the reference value falls short, since, in a typical day (one with variability in the stock value) the maximum value for the day must be higher than the reference value.

We have used our prediction system to generate and compare three alternative ways of generating sets of examples for each of the 5 stocks considered in this paper. In the first approach, that will be labeled as "Data Set 1" in the tables of results, we have used a direct embed of the time series we are predicting. Namely, we developed 25 features that represent the percent variations between the last 25 consecutive pairs of values of the reference values time series (this implies using information associated to the last known 26 values of each reference values time series). The second approach compared in our study (that we will refer to as "Data Set 2") uses a similar direct embed strategy but now considering several of the time series available for each stock. In this alternative we have also used 25 features, which include the 5 last known percent changes of the reference values time series, and the 4 last known percent changes of the volume, opening, maximum, minimum and closing values time series[5]. Finally, in the third alternative data preprocessing approach, that we will be labeled as "Data Set 3", we developed 25 features based on domain knowledge. This knowledge was used both to generate and to select the final set of features. Since there are literally hundreds of technical indicators applicable to our base data[6], the development of this type of features requires some restricting choices. We have used a process to choose the final set of features where some of them were fixed from the start, while others went through an automatic feature selection process. Among the former we included the first 4 terms of the simple embed used in the first data preprocessing approach. Regarding the automatic selection of the other features, we used the knowledge representation language to describe and generate a set of features that seemed relevant, and then used a feature ranking process[7] to discard those with lowest evaluations. In order to increase the statistical significance of this selection process we have used the combined 2000 training examples (400 training examples × 5 stocks).

It is interesting to analyze which were the features that resulted from the process described above. It was rather surprising to find out that most of the retained features where not the "full" technical analysis indicators but "constructive elements" of those indicators. An example is the retention of simple ratios of moving averages, instead of more complex variants of the Moving Average Convergence Divergence (MACD) indicator that where also considered as candidate features.

The final set of 25 chosen features can be grouped as follows:

- One feature based on the date time series, stating the weekday of the example.
- Six features based on the reference values time series, four of them embeds of the differences, the other two relating moving averages of different lengths.
- Three features based on direct embeds of differences of the BVL30 time series.
- Two features based on the differential changes of the reference values and BVL30 time series, during two different time frames.
- Three features based on the volume of trade time series, one of them an embed of the differences, the other two relating moving averages of different lengths.
- Five features based on different relations of the opening, maximum, minimum, closing and reference values of the last known day.
- Five features also based on relations of the opening, maximum, minimum, closing and reference values, but using moving averages of different lengths.
-

Overall, among the 25 developed features, the oldest historical values used are reference values up to 20 days old and traded volumes up to 15 days old.

In order to evaluate the prediction performance of the three alternative ways of pre-processing the raw stock data we used two common prediction error measures [10],

[5] With the help of our domain knowledge representation language it is quite simple to represent these two first sets of features. The first is represented by the single expression *percent(ref(i),ref(i+1))* with *i=0..24*. The second is represented by 6 very similar expressions, one for each of the 6 base data time series involved.

[6] Many of them involving a choice of parameters and thus greatly increasing the number of possible variants.

[7] This process is similar to the one used in the data mining component of our system described in Section 3.

chosen for their relevance for trading criteria evaluation. The first is the percent accuracy of the binary (rise or fall) predictions for the next day, defined as,

$$\%Acc = \frac{1}{N_{test}} \sum_{i=1}^{N_{test}} Score(y_i, \hat{y}_i) \tag{2}$$

where, N_{test} is the number of test cases; y_i is the truth target value of test case i; $\hat{y}_i$ is the predicted value for that test case; and

$$Score(y, \hat{y}) = \begin{cases} 1 & \text{if } y \geq 0 \wedge \hat{y} \geq 0 \\ 1 & \text{if } y < 0 \wedge \hat{y} < 0 \\ 0 & \text{otherwise} \end{cases}$$

The second evaluation measure we have used is the average value of the return on the test examples, taken as positive when the prediction sign is correct and as negative when it is wrong[8]. We called this measure the Average Return on the Predicted Examples (ARPE), which can be defined as,

$$ARPE = \frac{1}{N_{test}} \sum_{i=1}^{N_{test}} \left| Ref_i - Ref_{i-1} \right| sign(y_i \hat{y}_i) \tag{3}$$

where, N_{test} is the number of test cases; y_i is the truth target value of case i; $\hat{y}_i$ is the predicted value for that case; Ref_i is the reference value of case i.

Tables 1 and 2 show the accuracy and ARPE results obtained in our experiments using the Lazy3 and PA6 algorithms over the three data sets generated with the alternative pre-processing methods, which were described earlier in this Section. These tables also include the results obtained with two traditional benchmarks: the Naïve Prediction of Returns (NPR); and the buy-and-hold strategy[9]. The NPR merely predicts that the next percentage change of the reference value (the return of this value), will be the same as the last known change (*i.e.* $\hat{y}(t+1) = y(t)$). The results for this benchmark are presented in Table 1 as they are directly comparable with the other scores shown on this table. The results for the buy-and-hold benchmark are show in Table 2 as the average gain of the buy-and-hold strategy[10] over the 400 trading days involved in our tests, so that they are directly comparable with the other ARPE measure values presented.

The results shown on these tables provide strong empirical evidence towards our hypothesis concerning the use of domain knowledge to generate the training examples. In effect, considering the results over the 5 stocks, we can observe that the models obtained with data set 3 (that contain the features obtained with domain knowledge), consistently achieve better accuracy results than the others, independently of the algorithm used to obtain the model. When comparing the results

[8] Notice that the target values of each example are the percent changes between the last known reference value and the reference value of the next day.

[9] An extended discussion of these benchmarks can be found in [10] and [21].

[10] This strategy consists of buying at the price of last known training case, Ref_{50+400}, and selling at the price of the last known reference value $Ref_{50+400+400}$.

between data sets 1 and 2, we also observe worse results with the simple embeds (with a single exception in the BCP stock using the PA6 algorithm), than with embeds of multiple time series. If we consider the ARPE evaluation measure the conclusions are similar. An interesting point to notice is the fact that all the algorithm / data sets combinations produce positive results (accuracy results greater than 50% and ARPE results greater than 0).

Table 1. Percent accuracy over the test examples

	Lazy3 algorithm			PA6 algorithm			NRP
	Data Set 1	Data Set 2	Data Set 3	Data Set 1	Data Set 2	Data Set 3	
BCP	57.50	62.75	69.00	56.50	56.50	62.75	65.50
Brisa	54.75	62.00	66.50	57.75	63.25	63.75	56.75
Cimpor	59.75	63.00	69.25	56.00	57.00	64.50	58.50
EDP	58.50	65.75	72.25	58.00	67.75	72.50	59.00
PT	57.50	61.75	68.50	57.50	63.50	67.75	57.50
Average	57.60	63.05	69.10	57.15	61.60	66.25	59.45

Table 2. Average Return on the Predicted Examples (ARPE)

	Lazy3 algorithm			PA6 algorithm			Buy-and-hold
	Data Set 1	Data Set 2	Data Set 3	Data Set 1	Data Set 2	Data Set 3	
BCP	0.134	0.253	0.286	0.193	0.215	0.215	0.002
Brisa	0.176	0.286	0.394	0.228	0.312	0.333	0.099
Cimpor	0.266	0.404	0.498	0.192	0.386	0.447	0.141
EDP	0.263	0.515	0.578	0.302	0.531	0.591	0.010
PT	0.494	0.762	1.045	0.482	0.922	1.007	0.113
Average	0.267	0.444	0.560	0.279	0.473	0.519	0.073

Analyzing the benchmark results in isolation, we notice that the NPR accuracy results are considerably above the 50% average, which is somehow unexpected. This is due to the high auto-correlation (at lag 1) of the five time series in analysis (which also helps to explain the prediction ability of data set 1, as used in our system)[11]. Comparing our results with those achieved by the benchmarks, we notice that all the algorithm / data sets combinations achieve considerably better ARPE values than the buy-and-hold benchmark. On the other hand, the NPR accuracy results are globally higher than the results of the simple embed used in data set 1, although they are worse than the results of data set 2 and, particularly, of data set 3.

In order to assert the statistical significance of these results, we have carried out one-sided paired *t* tests over the 400 test examples of each stock, using the accuracy scores. The accuracy differences between models obtained with data set 1 and data set 3 were observed to be highly significant. In effect, with the Lazy 3 algorithm all the differences are statistically significant with a 99.5% confidence level, except for BCP where the level was 97.5%. Regarding the PA6 algorithm, the improvements obtained

[11] An analysis of those time series showed no significant auto-correlation at other lags, limiting the potential effectiveness of more powerful prediction techniques based in auto-correlation.

with data set 3 are somewhat less marked, but still significant with 99.5% confidence for Cimpor, EDP, and PT, with 97.5% confidence for BCP, and with 95% for Brisa.

5 Conclusions

This paper described an approach to financial time series prediction whose main distinctive feature is the use of domain knowledge to generate the training cases that form the basis of model construction. With this purpose we have developed a domain knowledge representation language that allows the time series expert to easily express his knowledge. The description obtained with the help of this language is then used by our prediction system to generate a training sample from the raw times series data.

We have contrasted this knowledge intensive approach with the traditional method of embedding the last time series values. With this purpose we have carried out a series of comparative experiments using time series data from five actively traded Portuguese companies. The results of our experiments with these stocks provide strong empirical evidence that a data preprocessing approach based on a direct embed of the time series has large limitations, and that the use of features constructed from the available raw data with the help of domain knowledge is advantageous.

Regarding future developments of this research we intend to extend the experimental evaluation to prediction horizons larger than the next day, as well as further refine the important feature selection phase.

References

1. Abu-Mostafa, Y., LeBaron, B., Lo, A. and Weigend, A. (eds.): Proceedings of the Sixth International Conference on Computational Finance, CF99. MIT Press (1999)
2. Almeida, P. and Bento, C.: Sequential Cover Rule Induction with PA3. Proceedings of the 10th International Conference on Computing and Information (ICCI'2000), Kuwait. Springer-Verlag (2001)
3. Bellman, R.; Adaptative Control Processes: A Guided Tour. Princeton University Press, (1961)
4. Bishop, C.: Neural Networks for Pattern Recognition. Oxford University Press (1995)
5. Bontempi, G.: Local Learning Techniques for Modeling, Prediction and Control. Ph.D. Dissertation, Université Libre de Bruxelles, Belgium (1999)
6. Box, G., and Jenkins, G.: Time Series Analysis, Forecasting and Control. Holden-Day (1976)
7. Demark, T.: The New Science of Technical Analysis. John Wiley & Sons (1994)
8. Fama, E.: Efficient Capital Markets: A review of Theory and Empirical Work. Journal of Finance, 25 (1970) 383-417
9. Hall, M.: Correlation-Based Feature Selection for Machine Learning. Ph.D. Dissertation, Department of Computer Science, University of Waikato (1999)
10. Hellstrom, T.: Data Snooping in the Stock Market. Theory of Stochastic Process 5(21) (1999)
11. Herbst, A.: Analyzing and Forecasting Futures Prices. John Wiley & Sons (1992)

12. Hong, S.: Use of Contextual Information for Feature Ranking and Discretization. IEEE Transactions on Knowledge and Data Engineering, 9(5) (1997) 718-730
13. Hutchinson, J.: A Radial Basis Function Approach to Financial Time Series Analysis. Ph.D. Dissertation, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology (1994)
14. Kustrin, D.: Forecasting Financial Time series with Correlation Matrix Memories for Tactical Asset Allocation. Ph.D. Dissertation, Department of Computer Science, University of York, UK (1998)
15. Lawrence, S., Tsoi, A. and Giles C.: Noisy Time Series Prediction using Symbolic Representation and Recurrent Neural Network Grammatical Inference. Technical Report UMIACS-TR-96-27 and CS-TR-3625, Institute for Advanced Computer Studies, University of Maryland, MD (1996)
16. Michalski, R.: A Theory and Methodology of Inductive Learning. In Michalski, R., Carbonell, J., and Mitchell, T., (eds): Machine Learning: An Artificial Intelligence Approach, Vol. 1. Morgan Kaufmann (1983)
17. Mozer, M.: Neural Net Architectures for Temporal Sequence Processing. In: Weigend, A. and Gershenfeld, N. (eds.): Time Series Prediction: Forecasting the Future and Understanding the Past. Addison-Wesley (1994)
18. Murphy, J.: Technical Analysis of the Financial Markets: A Comprehensive Guide to Trading Methods and Applications. Prentice Hall (1999)
19. Murthy, S., Kasif, S. and Salzberg, S.: A System for Induction of Oblique Decision Trees. Journal of Artificial Intelligence Research, 2 (1994) 1-32
20. Povinelli, R.: Time Series Data Mining: Identifying Temporal Patterns for Characterization and Prediction of Time Series Events. Ph.D. Dissertation, Marquette University, Milwaukee, Wisconsin (1999)
21. Refenes, A.: Testing Strategies and Metrics. In Refenes, A. (ed.): Neural Networks in the Capital Markets. John Wiley & Sons (1995)
22. Sauer, T., Yorke, J. and Casdagli, M.: Embedology. Journal of Statistical Physics 65 (1991) 579-616
23. Scott, D.; Multivariate Density Estimation. John Wiley & Sons (1992)
24. Takens, F.: Detecting Strange Attractors in Turbulence. In Rand, D. and Young, L. (eds.), Lecture Notes in Mathematics, Vol. 898. Springer (1981) 366-381
25. Weigend, A. and Gershenfeld, N. (eds.): Time Series Prediction: Forecasting the Future and Understanding the Past. Addison-Wesley (1994)
26. Weigend, A., Huberman, B. and Rumelhart, D.: Predicting Sunspots and Exchange Rates with Connectionist Networks. In Casdagli, M. and Eubank, S. (eds.): Nonlinear Modeling and Forecasting, SFI Studies in the Sciences of Complexity. Addison-Wesley (1992)
27. Weigend, A., Zimmermann, H. and Neuneier, R.: Clearning. In Refenes, P., Abu-Mostafa, Y., Moody, J. and Weigend, A. (eds.): Neural Networks in Financial Engineering (Proceedings of NNCM'95). World Scientific (1996)
28. Weiss, S. and Indurkhya, N.: Predictive Data Mining: A Practical Guide. Morgan Kaufmann (1998)
29. Yule, G.: On a Method of Investigating Periodicities in Disturbed Series with Special Reference to Wolfer's Sunspot Numbers. Phil. Trans. Royal Society, Series A, 226 (1927)
30. Yuret, D. and Maza, M.: A Genetic Algorithm System for Predicting de OEX. Technical Analysis of Stocks and Commodities, 12(6) (1994) 255-259
31. Zang, X. and Hutchinson, J.: Simple Architectures on Fast Machines: Practical Issues in Nonlinear Time Series Prediction. In Weigend, A. and Gershenfeld, N. (eds.): Time Series Prediction: Forecasting the Future and Understanding the Past. Addison-Wesley (1994)

Optimizing the Sharpe Ratio for a Rank Based Trading System

Thomas Hellström

Department of Computing Science
Umeå University, 901 87 Umeå, Sweden
thomash@cs.umu.se
http://www.cs.umu.se/~thomash

Abstract. Most models for prediction of the stock market focus on individual securities. In this paper we introduce a rank measure that takes into account a large number of securities and grades them according to the relative returns. It turns out that this rank measure, besides being more related to a real trading situation, is more predictable than the individual returns. The ranks are predicted with perceptrons with a step function for generation of trading signals. A learning decision support system for stock picking based on the rank predictions is constructed. An algorithm that maximizes the Sharpe ratio for a simulated trader computes the optimal decision parameters for the trader. The trading simulation is executed in a general purpose trading simulator ASTA. The trading results from the Swedish stock market show significantly higher returns and also Sharpe ratios, relative the benchmark.

1 Introduction

The returns of individual securities are the primary targets in most research that deal with the predictability of financial markets. In this paper we focus on the observation that a real trading situation involves not only attempts to predict the individual returns for a set of interesting securities, but also a comparison and selection among the produced predictions. What an investor really wants to have is not a large number of predictions for individual returns, but rather a grading of the securities in question. Even if this can be achieved by grading the individual predictions of returns, it is not obvious that it will yield an optimal decision based on a limited amount of noisy data. In Section 2 we introduce a rank measure that takes into account a large number of securities and grades them according to the relative returns. The rank concept has in a previous study [5] shown a potential for good predictability. In Section 4, perceptron models for prediction of the rank are defined and historical data is used to estimate the parameters in the models. Results from time series predictions are presented. The predictions are used as a basis for a learning decision support system for stock picking described in Section 5. The surprisingly successful results are discussed. Section 6 contains a summary of the results together with ideas for future research.

2 Defining a Rank Measure

The k-day return $R_k(t)$ for a stock m with close prices $y^m(1), ..., y^m(t_1)$ is defined for $t \in [k+1, ..., t_1]$ as

$$R_k^m(t) = \frac{y^m(t) - y^m(t-k)}{y^m(t-k)}. \tag{1}$$

We introduce a rank concept A_k^m, based on the k-day return R_k as follows: The k-day rank A_k^m for a stock s_m in the set $\{s_1, ..., s_N\}$ is computed by ranking the N stocks in the order of the k-day returns R_k. The ranking orders are then normalized so the stock with the lowest R_k is ranked -0.5 and the stock with the highest R_k is ranked 0.5. The definition of the k-day rank A_k^m for a stock m belonging to a set of stocks $\{s_1, ..., s_N\}$, can thus be written as

$$A_k^m(t) = \frac{\#\{R_k^i(t) | R_k^m(t) \geq R_k^i(t), 1 \leq i \leq N\} - 1}{N-1} - 0.5 \tag{2}$$

where the # function returns the number of elements in the argument set. This is as integer between 1 and N. R_k^m is the k-day returns computed for stock m. The scaling between -0.5 and $+0.5$ assigns the stock with the median value on R_k the rank 0. A positive rank A_k^m means that stock m performs better than this median stock, and a negative rank means that it performs worse. This new measure gives an indication of how each individual stock has developed relatively to the other stocks, viewed on a time scale set by the value of k.

The scaling around *zero* is convenient when defining a prediction task for the rank. It is clear that an ability to identify, at time t, a stock m, for which $A_h^m(t+h) > 0, h > 0$ means an opportunity to make profit in the same way as identifying a stock, for which $R_h(t+h) > 0$. A method that can identify stocks m and times t with a mean value of $A_h^m(t+h) > 0, h > 0$, can be used as a trading strategy that can do better than the average stock. The hit rate for the predictions can be defined as the fraction of times, for which the sign of the predicted rank $A_h^m(t+h)$ is correct. A value greater than 50% means that true predictions have been achieved. The following advantages compared to predicting returns $R_h(t+h)$ can be noticed:

1. The benchmark for predictions of ranks $A_h^m(t+h)$ performance becomes clearly defined:
   - A hit rate $> 50\%$, for the predictions of the sign of $A_h^m(t+h)$ means that we are doing better than chance. When predicting returns $R_h(t+h)$, the general positive drift in the market causes more than 50% of the returns to be > 0, which means that it is hard to define a good benchmark.
   - A positive mean value for predicted positive ranks $A_h(t+h)$ (and a negative mean value for predicted negative ranks) means that we are doing better than chance. When predicting returns $R_h(t+h)$, the general positive drift in the market causes the returns to have a mean value > 0. Therefore, a mere positive mean return for predicted positive returns does not imply any useful predicting ability.

2. The rank values $A_k^1(t), ..., A_k^N(t)$, for time t and a set of stocks $1, ..., N$ are uniformly distributed between -0.5 and 0.5 provided no return values are equal. Returns R_k^m, on the other hand, are distributed with sparsely populated tails for the extreme low and high values. This makes the statistical analysis of rank predictions safer and easier than predictions of returns.
3. The effect of global events gets automatically incorporated into the predictor variables. The analysis becomes totally focused on identifying deviations from the average stock, instead of trying to model the global economic situation.

3 Serial Correlation in the Ranks

We start by looking at the serial correlation for the rank variables as defined in (2). In Table 1 mean ranks $A_1^m(t+1)$ are tabulated as a function of $A_k^m(t)$ for 207 stocks from the Swedish stock market 1987-1997. Table 2 shows the "*Up fraction*", i.e. the number of positive ranks $A_1^m(t+1)$ divided by the number of non-zero ranks. Table 3 finally shows the number of observations of $A_1^m(t+1)$ in each table entry. Each row in the tables represents one particular value on k, covering the values $1, 2, 3, 4, 5, 10, 20, 30, 50, 100$. The label for each column is the mid-value of a symmetrical interval. For example, the column labeled 0.05 includes points with k-day rank $A_k^m(t)$ in the interval $[\ 0.00, ..., 0.10\ [$. The intervals for the outermost columns are open-ended on one side. Note that the stock price time series normally have 5 samples per week, i.e. $k = 5$ represents one week of data and $k = 20$ represents approximately one month. Example: There are 30548 observations where $-0.40 \leq A_2^m(t) < -0.30$ in the investigated data. In these observations, the 1-day ranks on the following day, $A_1^m(t+1)$, have an average value of 0.017, and an "*Up fraction*" = 52.8%.

Table 1. Mean 1-step ranks for 207 stocks

	k-day rank									
k	**-0.45**	**-0.35**	**-0.25**	**-0.15**	**-0.05**	**0.05**	**0.15**	**0.25**	**0.35**	**0.45**
1	0.067	0.017	-0.005	-0.011	-0.011	-0.004	-0.005	-0.010	-0.014	-0.033
2	0.060	0.017	0.002	-0.004	-0.010	-0.003	-0.007	-0.015	-0.017	-0.032
3	0.057	0.016	0.003	-0.005	-0.003	-0.008	-0.011	-0.011	-0.015	-0.034
4	0.054	0.018	0.003	-0.003	-0.005	-0.008	-0.011	-0.013	-0.012	-0.032
5	0.051	0.015	0.004	-0.002	-0.004	-0.009	-0.010	-0.009	-0.016	-0.032
10	0.040	0.013	0.005	-0.001	-0.003	-0.006	-0.007	-0.009	-0.012	-0.030
20	0.028	0.008	0.003	-0.003	-0.002	-0.002	-0.006	-0.011	-0.009	-0.019
30	0.021	0.007	0.002	0.004	-0.003	-0.003	-0.006	-0.006	-0.011	-0.015
50	0.014	0.005	0.000	-0.000	-0.001	-0.002	-0.005	-0.004	-0.006	-0.010
100	0.007	0.003	0.001	-0.002	-0.003	-0.004	-0.004	-0.004	-0.003	-0.008

Table 2. Fraction up/(up+down) moves (%)

	k-day rank									
k	**-0.45**	**-0.35**	**-0.25**	**-0.15**	**-0.05**	**0.05**	**0.15**	**0.25**	**0.35**	**0.45**
1	59.4	52.9	49.1	47.3	48.0	49.6	49.5	48.2	47.8	46.4
2	58.4	52.8	49.7	48.9	48.4	49.7	48.9	47.4	47.6	46.2
3	58.1	52.4	50.3	48.9	49.1	49.0	48.1	48.1	47.8	46.1
4	57.5	52.5	50.4	49.2	49.0	48.7	48.0	47.9	48.6	46.3
5	57.1	52.0	50.4	49.4	49.1	48.5	48.2	48.6	47.7	46.3
10	55.6	51.7	50.4	49.8	49.3	48.8	48.7	48.5	48.2	46.3
20	53.8	51.1	50.2	49.6	49.4	49.5	49.0	48.3	48.7	47.8
30	52.7	50.9	50.3	50.8	49.1	49.2	48.8	48.9	48.5	48.4
50	52.0	50.7	49.6	49.9	49.6	49.6	49.0	49.3	49.1	48.9
100	51.4	50.4	49.9	49.5	49.2	49.2	49.0	49.2	49.6	49.1

The only clear patterns that can be seen in the table are a slight negative serial correlation: negative ranks are followed by more positive ranks and vice versa. To investigate whether this observation reflects a fundamental property of the process generating the data, and not only idiosyncrasies in the data, the relation between current and future ranks is also presented in graphs, in which one curve represents one year. Figure 1 shows $A_1^m(t+1)$ versus $A_1^m(t)$. I.e.: 1-day ranks on the following day versus 1-day ranks on the current day. The same relation for 100 simulated random-walk stocks is shown in Figure 2 for comparison.

From Figure 1 we can conclude that the rank measure exhibits a mean reverting behavior, where a strong negative rank in mean is followed by a positive rank. Furthermore, a positive rank on average is followed by a negative rank on the following day. Looking at the "*Up fraction*" in Table 2, the uncertainty in these relations is still very high. A stock m with a rank $A_1^m(t) < -0.4$ has a positive rank $A_1^m(t+1)$ the next day in no more than 59.4% of all cases. However, the general advantages described in the previous section, coupled with the shown correlation between present and future values, do make the rank variables very interesting for further investigations. In [3] the observed mean reverting behavior is exploited in a simple trading system. The rank measure in the next section is used both as input and output in a model for prediction of future ranks.

4 Predicting the Ranks with Perceptrons

For a stock m, we attempt to predict the h-day-rank h days ahead by fitting a function g_m so that

$$\hat{A}_h^m(t+h) = g_m(I_t) \tag{3}$$

where I_t is the information available at time t. I_t may, for example, include stock returns $R_k^m(t)$, ranks $A_k^m(t)$, traded volume etc. The prediction problem 3

Table 3. Number of points

	k-day rank									
k	**-0.45**	**-0.35**	**-0.25**	**-0.15**	**-0.05**	**0.05**	**0.15**	**0.25**	**0.35**	**0.45**
1	30878	30866	31685	30837	30434	31009	31258	30539	30951	31550
2	30926	30548	31427	30481	30442	31116	31263	30435	30841	31675
3	30922	30440	31202	30404	30350	31146	31061	30449	30814	31697
4	30887	30315	31052	30320	30371	31097	31097	30328	30777	31776
5	30857	30293	30951	30275	30191	31049	31144	30254	30701	31816
10	30755	30004	30648	29958	30004	30875	30889	30155	30571	31775
20	30521	29635	30306	29591	29679	30560	30580	29836	30377	31692
30	30388	29371	30083	29388	29567	30349	30437	29652	30190	31503
50	30117	29006	29728	28979	29306	29876	30109	29236	29927	31159
100	29166	28050	28790	28011	28238	29015	29049	28254	29012	30460

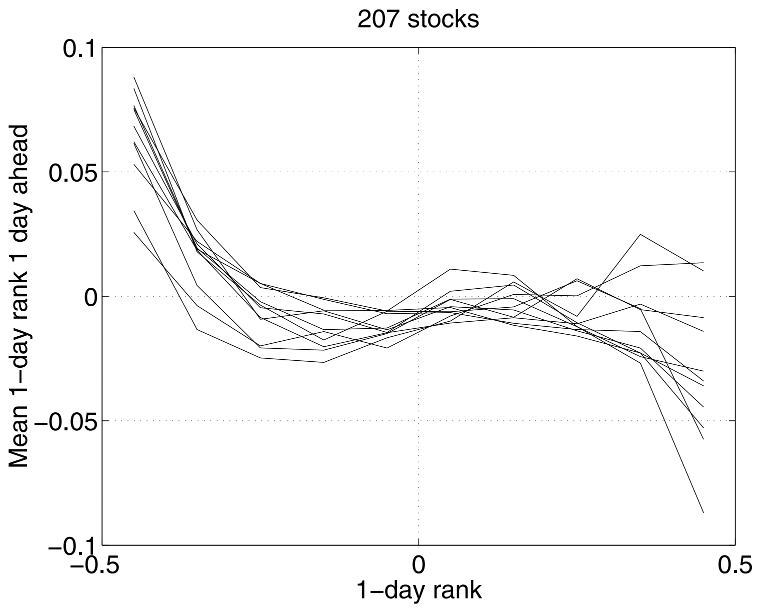


Fig. 1. 1-day ranks $A_1^m(t+1)$ versus $A_1^m(t)$. Each curve represents one year between 1987 and 1997.

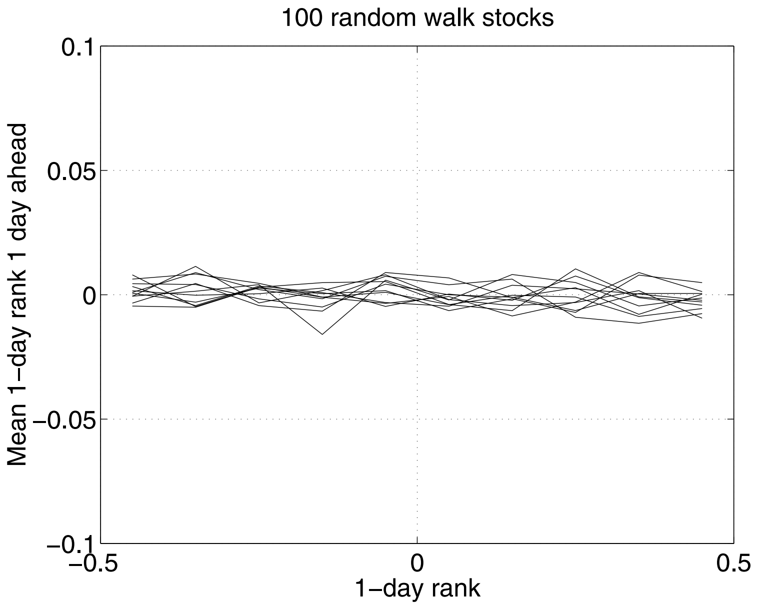


Fig. 2. 1-day ranks $A_1^m(t+1)$ versus $A_1^m(t)$. Each curve represents one year with 100 simulated random-walk stocks.

is as general as the corresponding problem for stock returns, and can of course be attacked in a variety of ways. Our choice in this first formulation of the problem assumes a dependence between the future rank $A_h^m(t+h)$ and current ranks $A_k^m(t)$ for different values on k. I.e.: a stock's tendency to be a *winner* in the future depends on its *winner* property in the past, computed for different time horizons. This assumption is inspired by the autocorrelation analysis in Hellström [5], and also by previous work by De Bondt, Thaler [1] and Hellström [3] showing how these dependencies can be exploited for prediction and trading. Confining our analysis to $1, 2, 5$ and 20 days horizons, the prediction model 3 is refined to

$$\hat{A}_h^m(t+h) = g_m(A_1^m(t), A_2^m(t), A_5^m(t), A_{20}^m(t)). \tag{4}$$

The choice of function g_m could be a complex neural network or a simpler function. Our first attempt is a perceptron, i.e. the model is

$$\begin{aligned}&\hat{A}_h^m(t+h) = \\ &f(p_0^m + p_1^m A_1^m(t) + p_2^m A_2^m(t) + p_3^m A_5^m(t) + p_4^m A_{20}^m(t))\end{aligned} \tag{5}$$

where the activation function f for the time being is set to a linear function. The parameter vector $p^m = (p_0^m, p_1^m, p_2^m, p_3^m, p_4^m)$ is determined by regression on historical data. For a market with N stocks, N separate perceptrons are built, each one denoted by the index m. The h-day rank A_h^m for time $t+h$ is predicted from the 1-day, 2-day, 5-day and 20-day ranks, computed at time t. To facilitate further comparison of the m produced predictions, they are ranked in a similar way as in the definition of the ranks themselves:

$$\begin{aligned}&\hat{A}_h^m(t+h) \leftarrow -0.5+ \\ &(\#\{\hat{A}_h^i(t)|\hat{A}_h^m(t) \geq \hat{A}_h^i(t), 1 \leq i \leq N\} - 1)\tfrac{1}{N}.\end{aligned} \tag{6}$$

In this way the N predictions $\hat{A}_h^m(t+h), m = 1,...,N$, get values uniformly distributed between -0.5 and 0.5 with the lowest prediction having the value -0.5 and the highest prediction having the value 0.5.

4.1 Data and Experimental Set-Up

The data that has been used in the study comes from 80 stocks on the Swedish stock market from January 1, 1989 till December 31, 1997. We have used a sliding window technique, where 1000 points are used for training and the following 100 are used for prediction. The window is then moved 100 days ahead and the procedure is repeated until end of data. The sliding window technique is a better alternative than cross validation, since data at time t and at time $t+k, k>0$ is often correlated (consider for example the returns $R_5^m(t)$ and $R_5^m(t+1)$). In such a case, predicting a function value $A_1^m(t_1+1)$ using a model trained with data from time $t > t_1$ is cheating and should obviously be avoided. The sliding window approach means that a prediction $\hat{A}_h^m(t+h)$ is based on close prices $y^m(t-k),...,y^m(t)$. Since 1000 points are needed for the modeling, the predictions are produced for the years 1993-1997.

4.2 Evaluation of the Rank Predictions

The computed models $g_m, m = 1,...,N$ at each time step t produce N predictions of the future ranks $A_h^m(t+h)$ for the N stocks. The N predictions $\hat{A}_h^m, m = 1,...,N$, are evenly distributed by transformation 6 in $[-0.5,...,0.5]$. As we shall see in the following section, we can construct a successful trading system utilizing only a few of the N predictions. Furthermore, even viewed as N separate predictions, we have the freedom of rejecting predictions if they are not viewed as reliable or profitable[1]. By introducing a cut-off value γ, a selection of predictions can be made. For example, $\gamma = 0.49$ means that we are only considering predictions $\hat{A}_h^m(t+h)$ such that $|\hat{A}_h^m(t+h)| > 0.49$.

The results for 1-day predictions of 1-day ranks $\hat{A}_1^m(t+1)$ for a $\gamma = 0.0$ and 0.49 are presented in Tables 4 and 5. Each column in the tables represents performance for one trading year with the rightmost column showing the mean values for the entire time period. The rows in the table contain the following performance measures:

[1] As opposed to many other applications, where the performance has to be calculated as the average over the entire data set.

1. *Hitrate*$_+$. The fraction of predictions $\hat{A}_h^m(t+h) > \gamma$, with correct sign. A value significantly higher than 50% means that we are able to identify higher-than-average performing stocks better than chance.
2. *Hitrate*$_-$. The fraction of predictions $\hat{A}_h^m(t+h) < -\gamma$, with correct sign. A value significantly higher than 50% means that we are able to identify lower-than-average performing stocks better than chance.
3. *Return*$_+$. 100·Mean value of the h-day returns $R_h^m(t+h)$ for predictions $\hat{A}_h^m(t+h) > \gamma$.
4. *Return*$_-$. 100·Mean value of the h-day returns $R_h^m(t+h)$ for predictions $\hat{A}_h^m(t+h) < -\gamma$.
5. #*Pred*$_+$. Number of predictions $\hat{A}_h^m(t+h) > \gamma$.
6. #*Pred*$_-$. Number of predictions $\hat{A}_h^m(t+h) < -\gamma$.
7. #*Pred*. Total number of predictions $\hat{A}_h^m(t+h)$.

All presented values are average values over time t and over all involved stocks m. The performance for the one-day predictions are shown in the Tables 4 and 5. In Table 4 with $\gamma = 0.00$, the hit rates *Hitrate*$_+$ and *Hitrate*$_-$ are not significantly different from 50% and indicate low predictability. However, the difference between the mean returns (*Return*$_+$ and *Return*$_-$) for positive and negative rank predictions shows that the sign of the rank prediction really separates the returns significantly. By increasing the value for the cut-off value γ to $\gamma = 0.49$, the hit rate goes up to 64.2.0% for predicted positive ranks (Table 5). Furthermore, the difference between the mean returns for positive and negative rank predictions (*Return*$_+$ and *Return*$_-$) is substantial. Positive predictions of ranks are in average followed by a return of 0.895% while a negative rank prediction in average is followed by a return of 0.085%. The rows #*Pred*$_+$ and #*Pred*$_-$ show the number of selected predictions, i.e. the ones greater than γ and the ones less than $-\gamma$ respectively. For $\gamma = 0.49$ these numbers add to about 2.7% of the total number of predictions. This is normally considered insufficient when single securities are predicted, both on statistical grounds and for practical reasons (we want decision support more often than a few times per year). But since the ranking approach produces a uniformly distributed set of predictions each day (in the example 80 predictions) there is always at least one selected prediction for each day, provided $\gamma < 0.5$. Therefore, we can claim that we have a method by which, every day we can pick a stock that goes up more than the average stock the following day with probability 64%. This is by itself a very strong result compared to most published single-security predictions of stock returns (see for example Burgess and Refenes [2], Steurer [8] or Tsibouris and Zeidenberg [9]).

5 Decision Support

The rank predictions are used as basis for a decision support system for stock picking. The layout of the decision support system is shown in Figure 3. The 1-day predictions $\hat{A}_1^m(t+1)$ are fed into a decision maker that generates buy and sell signals that are executed by the ASTA trading simulator. The decision maker

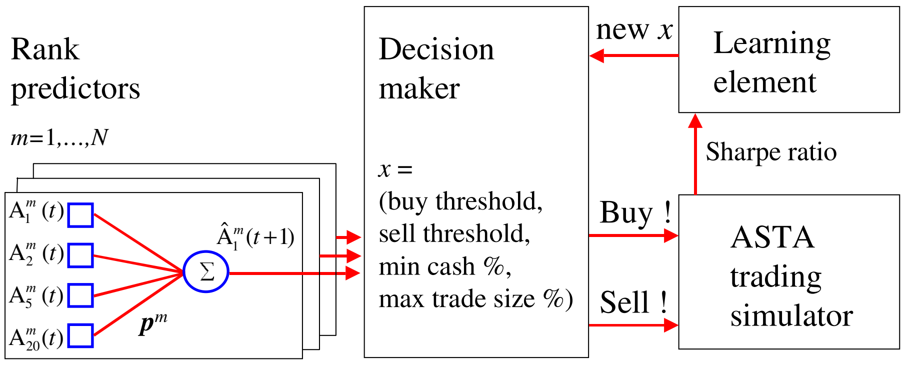


Fig. 3. The trading system is based on 1-day rank predictions for N stocks, and a decision maker parameterized by the x-vector. The learning element comprises a numerical optimizer, which finds the x that maximizes the Sharpe ratio for the historical data.

Table 4. 1-day predictions of 1-day ranks $|\hat{A}_1(t+1)| > 0.00$

$Year:$	**93**	**94**	**95**	**96**	**97**	**93-97**
$Hitrate_+$	51.1	53.4	53.3	53.0	52.5	52.7
$Hitrate_-$	51.8	53.6	53.4	53.2	52.6	52.9
$Return_+$	0.389	0.101	0.155	0.238	0.172	0.212
$Return_-$	0.253	-0.176	-0.094	0.057	0.008	0.010
$\#Pred_+$	7719	8321	8313	8923	8160	41510
$\#Pred_-$	7786	8343	8342	8943	8172	41664
$\#Pred$	15505	16664	16655	17866	16332	83174

Table 5. 1-day predictions of 1-day ranks $|\hat{A}_1(t+1)| > 0.49$

$Year:$	**93**	**94**	**95**	**96**	**97**	**93-97**
$Hitrate_+$	59.7	65.1	67.9	66.7	61.2	64.2
$Hitrate_-$	52.7	53.2	56.4	59.4	56.7	55.7
$Return_+$	1.468	0.583	0.888	0.770	0.745	0.895
$Return_-$	1.138	-0.236	-0.402	-0.040	-0.055	0.085
$\#Pred_+$	211	215	218	228	214	1088
$\#Pred_-$	222	220	220	234	217	1115
$\#Pred$	15505	16664	16655	17866	16332	83174

is controlled by a parameter vector x, comprising the threshold values of the step functions that generate buy and sell signals from the rank predictions. Two other parameters control the amount of money the trader is allowed to invest in one single trade, and how large a fraction of the total wealth should be kept in cash at all times. The learning element comprises an optimizing algorithm, which finds the parameter vector x that maximizes the mean annualized Sharpe ratio for the simulated trader. The learning period is 2 years and the found optimal x is then used to control the trading in the following year. The procedure is repeated yearly, using the last 2 years as learning period. The ASTA system is a general-purpose tool for development of trading and prediction algorithms. A technical overview of the system can be found in Hellström [4] and examples of usage in Hellström [3] and Hellström, Holmström [6]. More information can also be found at http://www.cs.umu.se/∼thomash. The rank measure and also the prediction algorithm described in Section 4 is implemented in ASTA and therefore the test procedure is very straightforward. A transaction cost of 0.15% (minimum 90 Swedish crowns ∼ USD) is assumed for every buy or sell order.

5.1 Trading Results

The annual trading profit is presented in Table 6. As can be seen, the performance is very good. The trading strategy outperforms the benchmark (the Swedish Generalindex) consistently and significantly every year and the mean annual profit made by the trading is 129.4%. The mean annual index increase during the same period is 27.4%. The Sharpe ratio which gives an estimate of a risk adjusted return shows the same pattern. The average Sharpe ratio for the trading strategy is 3.0 while trading the stock index Generalindex gives 1.6. By studying the annual performance we can conclude that these differences in performance is consistent for every year 1993-1997. Further more, the number of trades every year is consistently high (seven buy and seven sell per week), which increases the statistical credibility of the results. The trading results are also displayed in Figure 4. The upper diagram shows the equity curves for the trading strategy and for the benchmark index. The lower diagram shows the annual profits.

Table 6. Trading results for the trading system shown in Figure 3.

Year :	**93**	**94**	**95**	**96**	**97**	Mean	Total
Profit	120.8	116.7	142.9	165.8	100.8	129.4	6102
Index profit	52.1	4.6	18.3	38.2	23.8	27.4	222
Diff.	68.7	112.2	124.6	127.7	76.9	102.0	5880
#.trades	640	706	700	770	671	697	3487
Sharpe	1.7	2.4	3.2	4.5	3.1	3.0	
Index sharpe	2.8	0.1	1.4	2.4	1.4	1.6	

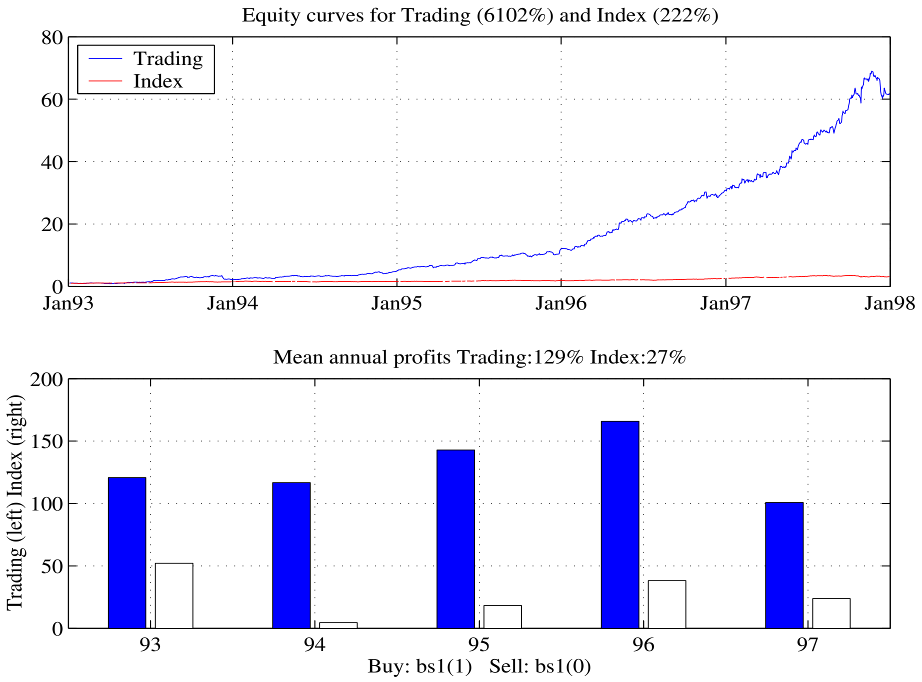


Fig. 4. Performance for the simulated trading with stock picking based on 1-day rank predictions as shown in Figure 3. The top diagram shows the equity curves while the lower diagram displays the annual profits. The trading outperforms the benchmark index consistently every year.

Let us look at possible reasons and mechanisms that may lie behind the good results. In [7], Lo and MacKinley report on positive *cross-autocovariances* across securities. These cross effects are most often positive in sign and are characterized by a *lead-lag* structure where returns for large-capitalization stocks tend to lead those of smaller stocks. Initial analysis of the trades that the rank strategy generates, expose a similar pattern, where most trading signals are generated for companies with relatively low traded volume. A positive *cross-autocovariances* can therefor provide part of an explanation to the successful trading results.

6 Conclusions

We have successfully implemented a model for prediction of a new rank measure for a set of stocks. The shown result is clearly a refutation of the Random Walk Hypothesis (RWH). Statistics for the 1-day predictions of ranks show that we are able to predict the sign of the threshold-selected rank consistently over the investigated 5-year-period of daily predictions. Furthermore, the mean returns that accompany the ranks show a consistent difference for positive and negative predicted ranks which, besides refuting the RWH, indicates that the rank concept could be useful for portfolio selection. The shown experiment with an optimizing trading system shows that this is indeed the case. The mean annual profit is 129.4% compared to 27.4% for the benchmark portfolio, over the investigated 5-year-period. The risk adjusted return, as measured by the Sharpe ratio, exhibits the same relation. The trading system gives a Sharpe ratio of 3.0 while trading the benchmark portfolio gives only 1.6.

Of course, the general idea of predicting ranks instead of returns can be implemented in many other ways than the one presented in this paper. Replacing the perceptrons with multi layer neural networks and also adding other kind of input variables to the prediction model (4) are exciting topics for future research.

References

1. W. D. Bondt and R. Taler. Does the stock market overreact? *Journal of Finance*, 40(3):793–805, 1985.
2. A. N. Burgess and A. N. Refenes. The use of error feedback terms in neural network modelling of financial time series. In C. Dunis, editor, *Forecasting Financial Markets*, Financial Economics and Quantitative Analysis, pages 261–274. John Wiley & Sons, Chichester, England, 1996.
3. T. Hellström. ASTA - a Tool for Development of Stock Prediction Algorithms. *Theory of Stochastic Processes*, 5(21)(1-2):22–32, 1999.
4. T. Hellström. ASTA - User's Reference Guide. Technical Report UMINF-00.16 ISSN-0348-0542, Department of Computing Science Umeå University, Umeå Sweden, 2000.
5. T. Hellström. Predicting a Rank Measure for Portfolio Selection. *Theory of Stochastic Processes*, 6(22)(3-4):64–83, 2000.
6. T. Hellström and K. Holmström. Parameter Tuning in Trading Algorithms using ASTA. In Y. S. Abu-Mostafa, B. LeBaron, A. W. Lo, and A. S. Weigend, editors, *Computational Finance 1999*, pages 343–357, Cambridge, MA, 1999. MIT Press.
7. A. W. Lo and A. C. MacKinley. *A Non-Random Walk Down Wall Street.* Princeton University Press, Princeton, 1999.
8. E. Steurer. Nonlinear modelling of the DEM/USD exchange rate. In A.-P. Refenes, editor, *Neural Networks in the Capital Markets*, chapter 13, pages 199–212. John Wiley & Sons, Chichester, England, 1995.
9. G. Tsibouris and M. Zeidenberg. Testing the efficent markets hypothesis with gradient descent algorithms. In A.-P. Refenes, editor, *Neural Networks in the Capital Markets*, chapter 8, pages 127–136. John Wiley & Sons, Chichester, England, 1995.

Choice: The Key for Autonomy

Luis Antunes, João Faria, and Helder Coelho

Faculdade de Ciências, Universidade de Lisboa
Campo Grande, 1749-016 Lisboa, Portugal
{xarax@di., faria@, hcoelho@di.}fc.ul.pt

Abstract. Agents must decide, i.e., choose a preferred option from among a large set of alternatives, according to the precise context where they are immersed. Such a capability de¨nes to what extent they are autonomous. But, there is no one way of deciding, and the classical mode of taking utility functions as the sole support is not adequate for situations constrained by qualitative features (such as wine selection, or putting together a football team). The BVG agent architecture relies on the use of values (multiple dimensions against which to evaluate a situation) to perform choice among a set of candidate goals. In this paper, we propose that values can also be used to guide the adoption of new goals from other agents. We argue that agents should base their rationalities on choice rather than search.

1 Introduction

For more than a decade, agent programming was based upon the BDI (Belief-Desire-Intention) model [12], and backed by Bratman's human practical reasoning theory [6]. However, this framework is not completely clari¨ed and extended for taking motivations and open situations into account. When looking deeply into the autonomy issue, we were faced with decision making at large, within non-deterministic contexts, because the BDI idea was not able to allow the implementation of a sound choice machinery. As a matter of fact, this issue is innovative from a cognitive science point of view, and our proposal based on values allows for a richer alternative to the so-called classical utility functions.

From a cognitive modelling standpoint, we bring the issue of choice into the nucleus of the agent architecture, and provide the mechanisms for adaptation of the choice machinery. From a decision theory standpoint, we propose a choice model that expands on the notion of utility to tackle multiple dimensions, and dynamic adaptation. From a philosophy of mind standpoint, we argue that evaluation of the results of choice cannot be made in absolute terms (the *perfect choice* does not exist), and so rationality is individual, situated and multi-varied. From a social simulation standpoint, we provide mechanisms to regulate (and later observe) interactions, namely, rules for goal adoption based on curiosity and imitation.

In [1, 3], we have proposed a new model of rationality that goes beyond utility and encompasses the notion of value as a central component for decision-making

Table 1. Comparison between the utilitarian and the multiple-values approaches.

Utility functions	Choice functions
one dimension	several dimensions
utility	values
closed world	open world
linear orderings	quasi-orderings
beforehand computation	execution-time computation

and computing importance. In this model, an agent has its cognitive machinery roughly included in a BDI framework, and produces its choices from available alternatives by considering, for the given situation, how each relevant value is affected. A calculus performs choices by collapsing all these partial assessments into a sorted list of actions to be performed. This must be made in execution time, since it is not possible to foretell all relevant possibilities beforehand. Evaluative assessments can evolve, as values are dynamic objects in the mind, and so can be added, removed, or changed. This framework does not add up to multi-attribute utility theory [10], anymore than it does to classical utility: the conditions for both these two choice theories are far too demanding, namely all alternatives must be comparable, and transitivity (or coherence) must be observed.

All these operations are favoured by the contact of the agent with the environment, including other agents, or some user. If we want to achieve enhanced adaptability to a complex, dynamic environment, we should provide the agent with motivations, rather than plain commands. A degree of autonomy is requested, as well as responsibility and animation. Autonomy is a social notion, and concerns the influence from other agents (including the user). An autonomous agent should be allowed to refuse some order or suggestion from another agent, but it should be allowed to adopt it as well.

In [3], the value-based adaptive calculus provided some interesting results when facing decision problems that called for some adaptability, even in the absence of all the relevant information. However, there remains one difficulty in assessing the results of those experiments: how do we evaluate our evaluations? We need meta-values with which to assess those results, but this calls for a designer, and amounts to look for emergent phenomena. It can be argued that if these 'higher' values exist why not to use them for decision? This dilemma shows clearly the ad hoc character of most solutions, and the difficulty in escaping it.

We can conceive two ways out. The first is the development of an ontology of values, to be used in some class of situations. Higher or lower, values have their place in this ontology, and their relations are clearly defined. For a given problem the relevant values can be identified and used, and appropriate experimental predictions postulated and tested.

The second is the object of this paper. By situating the agent in an environment with other agents, autonomy becomes a key ingredient, to be used with care and balance. The duality of value sets becomes a necessity, as agents cannot access values at the macro level, made judiciously coincide with the designer values. The answer is the designer, and the problem is methodological. The BVG (Beliefs, Values and Goals) model update mechanism provides a way to put to test this liaison between agent and designer. And, it clari¨es de¨nitely one of the mysteries of the BDI model, usually translated in an ad hoc way when some application is implemented.

In the next section we shed some light on the notion of choice, and argue for the introduction of motivational concepts di¨erent from the ones usually considered in BDI (i.e. desires). Decision is about choice and preference, and not about plans and search. Next we introduce the BVG architecture, and its related mechanisms for decision. In the fourth section we sustain that the multiple values decision framework will produce agents with enhanced autonomy, and present some value-based modes for regulating agent interaction. Then, we deal with one of these modes, by presenting concrete mechanisms for goal adoption. Section 6 discusses some forms of orderings we can use for the serialisation of candidate options. In section 7 and 8 we introduce our main case study, and discuss some of the experimental results we obtained. Finally we conclude by arguing that choice is one of the main ingredients for autonomy.

2 Departing from BDI

> "Practical reasoning is a matter of weighing con¨icting considerations for and against competing options, where the relevant considerations are provided by what the agent desires/values/cares about and what the agent believes." [6, page 17]

Bratman's theory of human practical reasoning was the inspiring set of ideas of the most used model of agency till today, the BDI model, where the trade-o¨ is between two distinct processes: deciding what (deliberation) versus deciding how (means-ends reasoning). Yet, the aim of the overall mechanism is directed towards actions, agent autonomy is out of the argumentation. So, the decision-making activity is not faced and its choosing stage not fully understood. We are ¨rmly convinced that without discussing how "competing options" are ranged, ordered, and selected, autonomy is no longer possible. Our thesis is the following: any agent uses values to decide and ¨nally execute some action. So, we need a theory of *human practical decision* to help us design and implement such autonomous agents.

Looking back to the ¨rst implementation of the BDI model, IRMA and PRS architectures (cf. respectively [7, 13]), we found a simple control loop between the deliberation and the means-ends reasoning processes. Further enhancements focused the deliberation process, the commitment strategies, the intention reconsideration [15], and the introduction of a belief revision function, a meta-level control component, or a plan generation function.

However, if we take a mainstream account of decision making such as [14], we ¨nd four stages involved: information gathering, likelihood estimation, deliberation (pondering of the alternatives) and choosing. If we look at Wooldridge's [15] descriptions of both deliberation (deciding what to do) and means-ends reasoning (deciding how to do it), we will discover that both are decision processes, and so can be encompassed by the above account. Studying the famous Cohen and Levesque's equation, Intention = Choice + Commitment, we discovered that the choice part was no so much dissected when compared to the commitment one, and no good reasons were found. BDI is a logician's dream that such processes can be 'solved' by theorem proving, or search, but those are not problems to be solved, rather decisions to be made. Or, broadly speaking, the issue at stake is not search, but rather *choice*. Search would imply the optimisation of a (one-dimensional) function, whereas in choice we can have a weighing over several dimensions (n-dimensional valuing).

The whole decision process is a mix of quantitative and qualitative modes of reasoning: ¨rst, a list of sub-goals is built by deliberation, and without any order; second, a partial order is imposed by weighing and valuing con¨icting considerations (e.g. feelings); third, the ¨rst sub-goals attached with values are "organised" unconsciously before the mind makes itself up. Recent ¨ndings of neuropsychology have come to the conclusion that many of our actions and perceptions are carried out by the unconscious part of our brains, and this means that emotion is no longer the sole control apparatus of decision-making [5].

The BVG (Beliefs, Values, Goals) model relaxes desires and intentions (gathered together into a general class of goals), puts out consciousness considerations, and only asserts that emotions are part of an overall global control machinery. We prefer, for simpli¨cation reasons, that the agent is in control of its own mind (cf. [8]). Making these choices, we can study what a¨ects the decision-making process (cf. ¨gure 1).

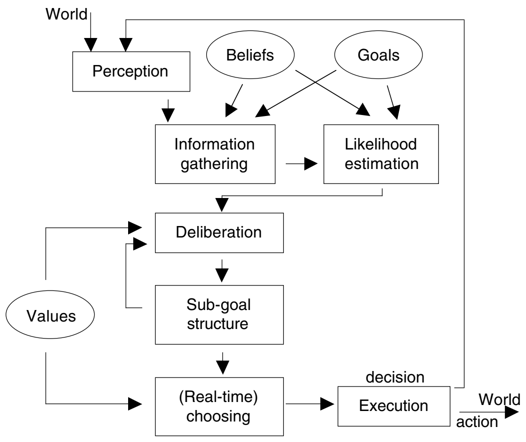


Fig. 1. Decision-making with the BVG architecture. Rectangles represent processes, and ovals represent sets with the main mental concepts. Arrows depict the ¨ow of data through the decision process.

Our work has mostly addressed the deliberation (in the sense of pondering of alternatives, since we regard 'BDI-deliberation' as a decision process in itself) part of the decision process. However, there is something to be said about the choosing stage. Once a sorting of the alternatives achieved, choosing can be as simple as to pick the ¨rst of the list, but other heuristics can be used, such as using an adequate probability distribution to pick one of the n ¨rst options, or again using values to inform this new process.

The agent's character is closely associated with his ability to choose the best/most appropriate sub-goal, i.e., each agent (egotistic, laconic, deceiving, etc.) has a speci¨c policy for choosing/preferring some goal. The so-called personality of an agent means heuristics for re-ordering options for arranging con¨gurations (clusters) of preferences, in a n-dimensional space.

The agent prefers an action by selecting credibility values (set-of-support) and by computing choice functions (policies), even amalgamating similar options (sub-goals) before a decision comes out.

3 The BVG Architecture

The BVG architecture roughly follows Castelfranchi's principles for autonomy contained in his "Double Filter Architecture" [8]. The reference schema of the BVG architecture for decision-making includes goals, candidate actions to be chosen from, beliefs about states of the world, and values about several things, including desirability of those states. Values are dimensions along which situations are evaluated, and actions selected. By dimension we mean a non empty set endowed with an order relation.

In the model proposed in [1], emotions have a determinant role in the control of the choice process. One main role of emotions is to set the amount of time available for a decision to be made. We proposed a cumulative method, that improves the quality of the decision when time allows it. The idea is based on the di¨erent importance of the other relevant values. Options are evaluated against the most important values ¨rst.

The other key issue in the BVG architecture is the update of the choice machinery based on assessments of the consequences of the previous decisions. These assessments can be made in terms of (1) some measure of goodness (the quality of the decision, measured through its consequences on the world); (2) the same dimensions that the agent used for decision; and (3) a di¨erent set of dimensions, usually the dimensions that the designer is interested in observing, what amounts to look for emergent behaviour (cf. [3]).

Issues (1) and (2) were addressed in [3], where a cumulative choice function $F = \sum c_k F_k$ was proposed, that, given some goal and alternatives characterised by values V_k, would sort those alternatives out, selecting the best of them for execution. Afterwards, the result of this decision gets evaluated and is fed back into the choice mechanism. An appropriate function performs updates on the features that are deemed relevant for the decision. For case (1), function G takes an assessment in dimension V_n, and distributes credit to every v_{ki}, feature of winning alternative i. In case (2), a more complex function H takes a multi-dimensional assessment in V_1 ♠ ♠♠♠♠ V_n. In both cases, the choice function F remains unchanged, since we sustain that our perception of alternatives is more likely to change than the way we perform choice. For instance, if based on some price-evaluating function P, we pick some wine for dinner and it turns out to be bad, we will probably prefer to mark that wine as a bad choice than to change P. P can no doubt change, but more slowly, and as a result of other factors, like the perception that we constantly pick bad wines, or a major increase in our income.

Case (3) relates the evaluation of the experiments to the behaviours of the agents. We have a keen concern for autonomous behaviours. But autonomy can only be located in the eyes of the observer. To be autonomous, some behaviour need not be expected, but also not extravagant. This hard-to-de¨ne seasoning will be transported into our agents by an update function such as described above, but one that doesn't depend exclusively on agent-available values. Returning to the wines example, suppose someone invites us over to dinner. A subservient friend would try to discover some wine we know and like to serve us with dinner. An autonomous one would pick some wine of his preference, regardless of the fact we don't drink any alcohol. In between of these two extreme behaviours is the concept of autonomy we are looking for. Our friend knows what we like, and serves something close but new, knowing we will cherish his care. A similar point was made in [4], where the opinions of a group of experts are to be combined. The opinion of each expert receives an a priori weight, and each expert aims at increasing his weight by being right most times about the outcome of the decision. The proven optimal strategy is to keep honesty and report one's true opinion, even if locally it might appear as the worst choice.

4 Autonomy in BVG

We propose several steps in this move towards enhanced autonomy. First, let us observe that adaptation and learning should be consequences of the interaction between the agent and the world and its components. If the agent would adapt as a result of any form of orders from its designer, it wouldn't be adaptation or learning, but design. Of course, if the agent is to respond to a user, this user should be present in its world. Castelfranchi's [8] postulates emphasise the role of social interactions in the agent's autonomy. We draw on these postulates to introduce some of the mechanisms involved in agent interaction. For further work on these issues, cf. [2].

Agents can share information, in particular evaluative information. Mechanisms to deal with this information can be ¨t into BVG in two di¨erent ways. (1) An agent receives information from another and treats it as if it was an assessment made by himself (possibly ¨ltered through a credibility assignment function). Choice can be performed as always, since the new information was incorporated into the old one and a coherent picture is available. (2) The receiving agent registers the new information together with the old one, and all is considered together by the choice function F when the time comes. A new component of F, say F_{n+1}, deals with the 'imported' information separately.

Agents can also share goals. They can pick goals from each other for several reasons. We try to characterise those reasons by recurring to values. We have addressed the adoption of goals by imitation (see [1]). Other reasons could be: curiosity (the appeal of something new in the absence of foreseen drawbacks); a¨ect (the adoption of some goal just because it pleases (or serves) another agent); etc. In section 5 we provide the mechanisms for some of these cases.

Finally, agents can share values. In humans, the acquisition of these takes years, and relies heavily on some ingredients: some 'higher' value, or notion of what's good and what's bad, that, by the way, we would postulate as a common element to any ontology of values; intricate dedicated mechanisms, that include some of the ones we have been addressing, but also more di¨ cult ones like kid's playing and repeating (as opposed to usual computer-like 'one-shot comprehension' [11]). Our mechanisms for value sharing are necessarily simple, and consist basically of conveying values and respective components of the choice and update functions. Decisions to adopt are arbitrary, as we are at this point more interested in looking at the e¨ects of this operations in the performance of the agents and their system.

5 Goal Adoption

Castelfranchi defends an agent should adopt a goal only if this new goal serves an already existent goal of his [8]. In [1] this idea is embodied into a goal adoption rule and expanded into value-based goal adoption. The general mechanism introduced was that of imitation. A possible goal is perceived and considered for adoption according to the agent's values. More complex rules were produced, that were based on values as the links between goals of the di¨erent agents.

In the case we wanted to maintain Castelfranchi's rule, our rule of adoption by imitation is to adopt the candidate goal if there would be already a goal that shared some value with the new goal to adopt. In the absence of a goal to be served by the goal candidate for adoption, we propose another rule that would base adoption upon the values concerning the imitated agent:

$$\begin{aligned}
&\mathit{Adopt}(agA,\ \mathit{Goal}_{((V_1,\omega_1),\ldots,(V_k,\omega_k))}(agB, G_0))\ \ \mathit{if}\\
&\qquad \exists \mathit{Bel}(agA,\ \mathit{Val}(agB, ((V_i,\ \frown_i), \ldots, (V_j,\ \frown_j)))) :\\
&\qquad\qquad \exists\, \mathit{Val}(agA, ((V_i,\ \urcorner_i), \ldots, (V_j,\ \urcorner_j))) :\\
&\qquad\qquad\qquad \forall m : i \spadesuit m \spadesuit j, \mathit{sameValue}(V_m,\ \frown_m,\ \urcorner_m) \qquad (1)
\end{aligned}$$

In this paper, we use a slightly di¨erent rule of adoption by imitation, by adapting Castelfranchi's requirement, while weakening the ¨nal condition linking the two goals, i.e., the values of agB are known through the description of its goal G_0, and it su¨ ces for agA to have one value coincident with the ones of agB

(we could have used *agA*'s goals to infer its values, but con¨icts might occur):

$$\begin{aligned} &Adopt(agA, Goal_{((V_1,\omega_1),\ldots,(V_k,\omega_k))}(agB, G_0)) \ \ if \\ &\qquad\qquad\qquad \exists\, Val(agA, (V_a, \frown_a)) : \\ &\qquad\qquad\qquad\quad \exists i (= a) \in \{1, \ldots, k\} : sameValue(V_i, \omega_i, \frown_a) \end{aligned} \quad (2)$$

Now we introduce another mechanism for value-based adoption: curiosity. An agent will adopt a goal from another agent if all of the values involved in this goal are completely new to him:

$$\begin{aligned} &Adopt(agA, Goal_{((V_1,\omega_1),\ldots,(V_k,\omega_k))}(agB, G_0)) \ \ if \\ &\qquad\qquad\qquad \neg\exists a : [1 \spadesuit a \spadesuit k \wedge Val(agA, (V_a, \frown_a))] \end{aligned} \quad (3)$$

We end this section by clearing up what the predicate sameValue means in the present framework. Given that most of our value features are numbers in a bounded scale, we set as standards the mean point of that scale (in cases were the scale is unbounded (V_4, see section 7) we postulated a 'normal' mean value). Agents de¨ne themselves with respect to each value by stipulating on which side of that standard they are in the respective scale. So, in a scale from 0 to 20, with standard 10, two agents with preference values 12 and 15 will be on the same side (i.e., they share this value).

6 Orderings

In classical decision theory, we ¨lter the set of possible options through a utility function (U), and into a linear ordering [9]. In particular, options with the same utility become indistinguishable. In the set of options, an equivalence relation $\frown$ becomes implicitly de¨ned: $a \,\tilde{}\, b$ *i¨* $U(a) = U(b)$. But we want to distinguish a from b, or rather, give the agent the possibility of choosing one of the two options, based on its motivations. The agent's motivations cannot be substituted by the sole motivation of maximising (expected) utility.

The situations we want to address demand for orderings weaker than the classical linear one, whether our standpoint is a deterministic analysis, or when the states of the world are not fully known, and a uncertainty measure can be associated: a probability, that depends on the agent's parameters. It is very clear to us that all uncertainty is prone to probabilisation, and this puts us foundationally in a subjectivist perspective of probability. If this were not the case, we would be stuck either with incomplete models or in situations where we cannot quantify the uncertainty.

When we aim to serialise alternatives, we have several orderings on the different dimensions that are relevant to choice. These orderings are really quasi-orderings (i.e., re¨exive, transitive and non antisymmetrical) and may have dichotomy or not. With these quasi-orderings we can build clusters of alternatives. Inside each cluster we can consider linear orderings, but the ¨nal decision must consider the universe of the di¨erent clusters. The global ordering (of all alternatives) we aim to get is typically not total (doesn't have dichotomy, because di¨erent dimensions are not comparable) and not partial (doesn't have antisymmetry, to avoid the indistinguishability of equally 'scoring' options, as explained above).

When we group alternatives in clusters built out of di¨erent dimensions, we are in fact using a common process in decision. (No agent in a market really knows all alternatives, only those that are accessible to him.) Clusterisation allows for the decrease of complexity, by excluding alternatives from the search spaces. Thus, the decision problem can be decomposed in simpler sub-problems. But the problem of integrating the results of those sub-problems remains. This integration or amalgamation of the relevant options must be made in terms of a dihcotomic quasi-ordering, because two di¨erent options must be distinguished, so that a choice can be performed. But the arrival at this 'total' mixture of orderings happens only in execution-time, it does not represent an intrinsic (*a priori*) property of the agent. And this allows for the evolution of the choice machinery in its components.

7 Case-Study: Selecting Wines

Experimentation has been conducted along a case study whose increasing complexity demands for the successive introduction of our ideas. The general theme is selection of wines by some intelligent agent. In [3], the agent would characterise wines through ¨ve dimensions against which to evaluate them: (V_1) the rating of the producer; (V_2) the quality of a given region in a given year, (V_3) the price; (V_4) the window of preferable consumption; and (V_5) the quality of the given wine.

Choice would then be performed by calculating the result of a choice function $F = \sum_{k=1}^{5} c_k F_k$ for each of the alternatives, and then picking up the lowest score. When the wines get tasted, the results of each tasting are fed back into the agent's choice machinery, and the impressions of the agent on the respective wine are updated. These results can be a simple rating of the quality of the wine, or a more complex qualitative description of the impressions of the wine. The results of performing the update in either of these two ways (cases 2 and 3 in section 3) led to di¨erent results on the agent's selection procedure.

8 Results

Somewhat unrealistically, once our agent achieved a conclusion about the best decision to be made, it would stick unconditionally to this decision: provided stock availability, it would keep buying the same wine. The reasons for this behaviour are clear, and have to do with the experimental setting not allowing for enough openness in the world, i. e. there is not enough source of change. So the agent does not have enough reason to change its choice. This is not such a bad result in certain contexts, for instance, people tend to stick to the same brand of car through large periods of their lives. This issue is linked with that of cognitive consonance, in that the weight of the previous decision overwhelms the desire to experiment new options (and the burden of carrying out a new complete choice process).

The solution we adopted was to limit stock availability, and let agents compete for the best wines, the market imposes itself on the agents. Other alternatives (to be eventually addressed) would be to adapt the choice function F to accommodate some novelty factor. This would amount to consider a new value representing the fact that every time we buy some wine we have a little less will to buy it again. In fact this amounts to using the 'law of diminishing returns,' classical in decision theory. Another solution would be to consider the choice procedure not to produce a unique winning decision, but a set of winning decisions from which our agent takes a probabilistically informed pick (see ¨gure 1). Each option in this set has a weight that comes from its contribution for the global score of the set. With these weights we can build a random distribution.

A series of simulations addressed the issue of goal exchange. To minimise the number of processes to control, we followed the policy of when in lack of reasons to perform some action (taken from a class of similar actions), this selection is arbitrarily done (usually using random numbers with adequate distribution). In our simulation, an agent will drop a randomly selected goal once in every four times he is active. This would allow us to concentrate on the mechanisms of goal exchange, instead of those of goals mental management.

We took 26 agents, of which only 2 were endowed with some goals. In some cases all goals were dropped rather quickly, whereas in others all agents had virtually all goals. In the most interesting cases, goals start travelling through the society. An agent who originally had all the goals may end up with just two, although he has spread them around before loosing him. There's the risk of the society loosing goals, although that did not happen frequently. The bigger danger is of the society loosing sense. What was meant as a means of enhancing bonds (through the share of interests) in the society ended up causing chaos. A typical result after 1000 runs would be: agA, from his original goals $\{g_1, g_2, g_3, gd_4, gd_5, gd_6\}$ has goals $\{gd_6, g_3\}$ only, he dropped goals some 14 times, including gd_6 (so this goal was later adopted from another agent). Goal gd_4 was lost, although this never would happen for goals g_1, g_2, and g_3 which were originally also owned by agB. A total of 269 goals were dropped, and the whole society has now 76 goals, spread through the agents. Two agents have no goals, although all agents have had goals at some time.

When we cross these simulations with the choice-perform-update ones, we reach the conclusion that what were the reasons for the selected actions may have been lost, with a high probability. Agents are committed to selections but they no longer have access to the rationality that caused those commitments, and so will have trouble deciding whether or not to keep them. Even worse, this hazards the update process and undermines the whole choice model, since this model by and large depends on a feedback circle that is now broken.

It is apparent that the overlooking of grounded goal management was the cause for this problem. The focus on preference-based choice was too heavy, and a return to a more solidly built mental basis may be in need, before readdressing the choice issues. This basis may as well be provided by BDI models, because of their emphasis in the web of technical, well-founded and well-structured mental states and corresponding relationships, and also because they are the inspiring models of BVG. Nevertheless, this only means that the balance between goal management and goal selection should be more thoroughly addressed. In no way these results undermine the value of explicitly focussing on choice in the agent architecture context.

9 Conclusions

The main driving force behind the BVG architecture has always been to design new rationalities (cognitive agent idealisations) and to inquire into motivation: what makes us do what we do; what allows us to balance what we ought to do, with what we can do, with what we want to do. And even how do we rationalise a bad (for our own judgement) decision in order not to pain over it again and again. The BVG model is another way to surpass the usual di¨ culties associated with the use of normative models from economics and decision theory (maximisation of expected utility). It provides a framework where multiple values are used to compute practical (good enough) decisions (reasoning from beliefs to values, and after, to goals and actions), and then these values are updated according to the outcome of those decisions. In this and other recent papers we have thrived to show how this model produces enhanced autonomy in an environment where the multiplicity of agents causes high complexity and unpredictability, by paying more attention to the choice machinery besides optimisation. But, the topic multiple agents is now in focus and the experiments done so far helped to improve our ideas around a better BVG proposal.

Our model departs from BDI (where choice is done by maximisation), by relaxing desires, and stating that agents need reasons for behaviour that surpass the simple technical details of what can be done, how it can be done and when it can be done. These reasons (which we might call 'dirty,' as opposed to 'clean,' logical ones) are then the basis of the agent's character, and are founded on values as representatives of the multiple agent's preferences. With the experimentation herein reported, we concluded that the dropping of those 'clean' reasons was perhaps hasty, and even a preference-conscious agent must possess a sound basis of logical mental concepts on which to ground. It is then only natural to return again to BDI as a provider of such a machinery, and to build the multiple-value schema on top of it. Another conclusion is the necessity of goals on which to base behaviour. Values are not enough, even when they can cause goals to be adopted. A careful weaving is still in order. However, we have found that interesting behaviours can be obtained from our agents: we ¨nd independent behaviours, as well as leaders and followers; we can observe several degrees of adaptation to a choice pattern, depending on the parameters of the model; we can follow the spread of goals through a society of value-motivated agents. The agents' exploration of the space of possibilities is carried out di¨erently from what would happen with a search procedure (even if guided by heuristics) since is driven by the agents' preferences, and these are heterogeneous.

Situated decision making is choosing among several alternatives and preferences in ways that emulate human beings. Alternatives are actions in the world, that are selected according to preference rankings or probability judgements (n-space values) to ¨t agent behaviours to precise dynamic contexts. Such an aim is the key for autonomy.

Acknowledgements

The research herein reported was conducted with partial support from FCT, within research unit LabMAC. We would like to thank the anonymous reviewers for the careful reading of the paper.

References

1. Luis Antunes and Helder Coelho. Redesigning the agents' decision machinery. In Ana Paiva, editor, *A¨ective Interactions*, volume 1814 of *Lecture Notes in Arti¨cial Intelligence*. Springer-Verlag, 2000. Revised and extended version of the paper presented in the Workshop on A¨ect in Interactions (Towards a new generation of interfaces), Siena, October 1999.
2. Luis Antunes, João Faria, and Helder Coelho. Autonomy through value-driven goal adoption. In Maria Carolina Monard and Jaime Simão Sichman, editors, *International Joint Conference: 7th Iberoamerican Conference on Arti¨cial Intelligence/15th Brazilian Conference on Arti¨cial Intelligence (open discussion track proceedings)*. ICMC/USP, São Carlos, SP, Brasil, 2000.
3. Luis Antunes, João Faria, and Helder Coelho. Improving choice mechanisms within the BVG architecture. In Yves Lësperance and Cristiano Castelfranchi, editors, *Intelligent Agents VII*, Lecture Notes in Arti¨cial Intelligence. Springer-Verlag, 2000. Proceedings of ICMAS 2000 workshop (ATAL), to appear.
4. M. J. Bayarri and M. H. DeGroot. Gaining weight: A bayesian approach. In J. M. Bernardo, M. H. DeGroot, D. V. D. V. Lindley, and A. F. M. Smith, editors, *Bayesian Statistics 3*. Oxford University Press, 1988.
5. A. Bechara, H. Damasio, D. Tranel, and A. R. Damasio. Deciding advantageously before knowing the advantageous strategy. *Science*, 275, 1997. February, 28.
6. Michael E. Bratman. What is intention? In Philip R. Cohen, Jerry L. Morgan, and Martha E. Pollack, editors, *Intentions in Communication*, pages 15–32. The MIT Press, Cambridge, MA, 1990.
7. Michael E. Bratman, David J. Israel, and Martha E. Pollack. Plans and resource-bounded practical reasoning. *Computational Intelligence*, 4:349–355, 1988.
8. Cristiano Castelfranchi. Guarantees for autonomy in cognitive agent architecture. In Michael Wooldridge and Nick Jennings, editors, *Intelligent Agents: agent theories, architectures, and languages*, volume 890 of *Lecture Notes in Arti¨cial Intelligence*. Springer-Verlag, 1995. Proceedings of ECAI'94 workshop (ATAL).

9. Simon French. *Decision Theory: An Introduction to the Mathematics of Rationality.* Ellis Horwood Limited, Chichester, England, 1986.
10. Ralph L. Keeney and Howard Rai¨a. *Decisions with Multiple Objectives: Preference and Value Tradeo¨s.* John Wiley and Sons, 1976.
11. A. Monk. Cyclic interaction: a unitary approach to intention, action and the environment. *Cognition*, 68, 1998.
12. Martha Pollack and Marc Ringuette. Introducing the Tileworld: experimentally evaluating agent architectures. In Thomas Dietterich and William Swartout, editors, *Proceedings of AAAI'90.* AAAI Press, Menlo Park, CA, 1990.
13. Anand S. Rao and Michael P. George¨. Modeling agents within a BDI-architecture. In R. Fikes and E. Sandewall, editors, *Proceedings of the 2nd International Conference on Principles of Knowledge Representation and Reasoning (KR'91)*, pages 473–484, Cambridge, Mass., 1991. Morgan Kaufmann publishers Inc.: San Mateo, CA, USA.
14. Robert A. Wilson and Frank C. Keil, editors. *The MIT Encyclopedia of the Cognitive Sciences.* The MIT Press, Cambridge, MA, 1999. second printing.
15. Michael Wooldridge. *Reasoning about rational agents.* The MIT Press, Cambridge, MA, 2000.

Dynamic Evaluation of Coordination Mechanisms for Autonomous Agents

Rachel A. Bourne[1], Karen Shoop[1], and Nicholas R. Jennings[2]

[1] Department of Electronic Engineering
Queen Mary, University of London
London E1 4NS, UK
{r.a.bourne,karen.shoop}@elec.qmul.ac.uk

[2] Dept of Electronics and Computer Science
University of Southampton, Highfield
Southampton SO17 1BJ, UK
nrj@ecs.soton.ac.uk

Abstract. This paper presents a formal framework within which autonomous agents can dynamically select and apply different mechanisms to coordinate their interactions with one another. Agents use the task attributes and environmental conditions to evaluate which mechanism maximises their expected utility. Different agent types can be characterised by their willingness to cooperate and the relative value they place on short- vs long-term rewards. Our results demonstrate the viability of empowering agents in this way and show the quantitative benefits that agents accrue from being given the flexibility to control how they coordinate.

1 Introduction

Autonomous agents are increasingly being deployed in complex applications where they are required to act rationally in response to uncertain and unpredictable events. A key feature of this rationality is the ability of agents to coordinate their interactions *in ways that are suitable to their prevailing circumstances* [5]. Thus, in certain cases it may be appropriate to develop a detailed plan of coordination in which each of the participant's actions are rigidly prescribed and numerous synchronisation points are identified. At other times, however, it may be appropriate to adopt much looser coordination policies in which the agents work under the general assumption that their collaborators are progressing satisfactorily and that no explicit synchronisation is needed. What this illustrates is that there is no universally best method of coordinating. Given this fact, we believe agents should be free to adopt, *at run-time*, the method that they believe is best suited to their current situation. Thus, for example, in relatively stable environments social laws may be adopted as the most appropriate means of coordinating [10], whereas in highly dynamic situations one-off contracting models may be better suited [11], and in-between, mechanisms that involve the high-level interchange of participants' goals may be best [4].

To achieve this degree of flexibility, agents need to be equipped with a suite of *coordination mechanisms* (CMs) (with different properties and characteristics), be provided with a means of assessing the likely benefit of adopting the various mechanisms in the prevailing circumstances, and have the ability to select and then enact the best mechanism. Against this background, this paper develops and empirically evaluates a generic decision making model that agents can employ to coordinate flexibly. Specifically, we identify a number of potentially differentiating features that are common to a wide range of CMs, provide a decision-theoretic model for evaluating and selecting between competing mechanisms, and empirically evaluate the effectiveness of this model, for a number of CMs, in a suitably general agent scenario. This work builds upon the preliminary framework of [2], but makes the following advances to the general state of the art. Firstly, using a range of CMs, we show that agents can effectively evaluate and decide which to use, dependent on their prevailing conditions. Secondly, the evaluation functions associated with these CMs highlight the different types of uncertainty agents need to cope with and the different environmental parameters they need to monitor, in order to coordinate flexibly. Thirdly, we show that individual agent features such as their willingness to cooperate and the degree to which they discount their future rewards affects which CM they adopt.

The remainder of the paper is structured in the following manner. Section 2 outlines the key components of the reasoning model and introduces the exemplar coordination models we evaluate in this work. Section 3 describes the grid world scenario we use for our evaluation. Section 4 formalises the reasoning models. Section 5 describes the experimental results and analysis. Section 6 looks at related work in this area. Finally, in section 7 we draw our conclusions.

2 Coordination Mechanisms

Flexible coordination requires the agents to know both how to *apply* a given CM and how to *reason about* which mechanism to select. In the former case, an agent must have access to the necessary protocols for coordinating with other agents and/or the environment. In the latter case, an agent must be capable of evaluating and comparing the possible alternatives.

2.1 Protocols for Coordination

Coordination involves the interworking of a number of agents, subject to a set of rules. The specification of exactly what is possible in a particular coordination context is given by the coordination protocol [8]. Thus, such protocols indicate the parties (or roles) that are involved in the coordination activity, what communication flows can occur between these parties, and how the participants can legally respond to such communications. Here we refer to the instigator of the coordination as the *manager* and to the other agents that assist the manager as the *subordinates* (or subs).

For example, in the Contract Net protocol, the manager initiates a two-stage process whereby bids are requested and received, and then selected and subs are appointed. In a simpler mechanism, such as being commanded by a superior officer, a sub simply obeys the commands it receives. In all cases, however, the key point is that for each mechanism an agent supports, it must have the ability and the know-how to enact the protocol.

2.2 Evaluation of Mechanisms

A manager that is faced with a coordination task will have several potential CMs at its disposal. Each such mechanism requires a means of determining the expected value it will provide, which should be comparable with the others available. To this end, an e*valuation function* is needed. The value of a given CM may depend on many features including: the reward structure, the likely time of completion, and the likely availability of subordinates. Generally speaking, the more complex the coordination protocol and reward structure, the more complex the evaluation function. In particular, the more uncertainty that exists in the agent's ability to set up and enact a CM, the harder it is to evaluate its use accurately. Moreover, some of the parameters that are needed for evaluation are likely to vary from mechanism to mechanism. A final consideration is that the value of an agent's current situation may also depend on the state of the mechanism it has adopted. For example, an agent that has successfully recruited other agents may value its state more highly than one still engaged in the recruitment process. For all these reasons, evaluation functions need to be tailored to the specific CM they describe.

When subordinates are invited to assist in coordination, they too must assess the value of accepting. These valuations are typically less complex than those for the managers since the reward on offer and the completion time are generally declared, though in some cases a sub may also need to handle uncertainty. Subs also need to take into account whether accepting an offer would incur any additional cost, such as a penalty for dropping a commitment to another agent.

2.3 Sample Mechanisms

This section outlines the protocols and reward structures for the CMs considered in this work (their evaluation functions are left to section 4). Clearly this list is not exhaustive. Rather our aim is to incorporate specific exemplars that are typical of the broad classes of coordination techniques that have been proposed in the literature. Thus, the precise form of each mechanism is of secondary importance to its broad characteristics and performance profile. Moreover, additional mechanisms can be incorporated simply by providing appropriate characterisations and evaluation functions. Nevertheless we believe that the chosen mechanisms are sufficient for our main objective of demonstrating the efficacy of dynamically selecting CMs.

In this work, tasks are assumed to have several attributes: a minimum personnel requirement (*mpr*), a total requirement of agent effort (*effort*), and a *reward* that is paid to the managing agent when the task is accomplished. Thus a task that has *mpr* of 3 and *effort* of 6 may be realised by 3 agents each contributing 2 units, by 6 agents each contributing 1 unit, but not by 2 agents each contributing 3 units. It follows that tasks with a high *mpr* are likely to incur a delay before the necessary subs are recruited and all agents can start working on the task together. Here agents can only work on one task at a time. However, each agent has a default task ($mpr = 1$, $effort = 1$) that it puts on hold whenever it agrees to participate in a more complex one. The type of task and how many are generated can all be varied experimentally.

Asocial CM: This mechanism is used when a manager elects to perform a task alone; therefore there is no need to coordinate with any subs. The manager adopts the task, works on it and, ultimately, receives all the reward. This CM can only be used on tasks for which $mpr = 1$.

Social Law CM: A manager may elect to have the task performed by invoking a social law (SL) that has been agreed in advance by all the agents in the system[1]. For a task with $mpr = n$, the nearest $n-1$ other agents are commanded to work on the task with the manager. Because the social law extends to all agents, the subordinates cannot refuse to help. They simply assist until they are released from the task. The reward is then divided equally among all the participants. In our experimental setting, the location of subs was performed by a central coordinator allowing minimal set up delay and the prevention of multiple managers attempting to command subs at the same time. A truly distributed version is possible though would require a longer set up time.

Pot Luck CM: A manager that elects to use Pot Luck (PL) coordination, sets terms under which it is willing to pay subs on a piecemeal (step-by-step) basis. These terms are then offered to all agents in the direct vicinity (this is called "pot luck" since the manager makes no active effort to recruit subs in the hope that some are already present or wander by shortly). When the task is completed the manager receives the full reward. From the subordinate's point of view, it is occasionally offered "temporary" work for an indefinite period at a fixed rate; it either accepts and works on the task, or declines and ignores the offer. This CM is likely to be more successful when the environment is densely populated. But because the manager issues a blanket offer, it runs the risk of both over- and under-recruitment of subs. A sub can decommit from a PL task at any time at no penalty, keeping any reward it has already earned.

Contract Net CM: A manager that elects to use Contract Net (CN) coordination requests bids from other agents that submit their terms according to their current circumstances. The manager selects from among the bids received and sends out firm offers. An agent receiving an offer either accepts and works on the task, eventually receiving a reward based on its bid, or declines. The manager may thus fail to recruit sufficient subs in which case it repeats the

[1] The process of agreeing the social law is not considered in this work. In particular, this means that the associated costs of acheiving consensus are not factored into the cost of this mechanism.

request-bid-offer cycle. When the task is accomplished, the manager receives the whole reward but must pay off the subs with the agreed amounts. Again, under this CM a manager may under recruit if some agent declines its offer. Once a sub has accepted a task under this CM, it may later decommit though it will have to pay a penalty for doing so. If a manager abandons a task under this CM, it must pay off any subs it has recruited, according to their terms.

3 Grid World Scenario

The scenario involves a number of autonomous agents occupying a grid world (see [2] for more details). Tasks are generated randomly according to the experimental conditions and are found by the agents who must decide whether or not to take them on. Each agent always has a specific (default) task that may be carried out alone. Tasks are located at squares in the grid and each has an associated *mpr*, *effort* and *reward*. One unit of effort is equivalent to one agent working on the task at the square for one time step (provided sufficient agents work on it in total). When accepting a task, an agent must decide how to tackle it; if the *mpr* is greater than one, it must recruit other agents to assist it, according to the various CMs at its disposal. To simplify the evaluation functions, tasks persist until they have been achieved. A more realistic setting with deadlines on tasks will require that the evaluation functions be extended to take into account the possibility of a task not being completed in time.

The agents themselves have various features that can be parameterised to simulate different behavioural types. Each agent has a *willingness to cooperate* (*wtc*) factor which it uses when bidding for and evaluating tasks; when this factor is low ($wtc < 1$) the agents are *greedy* and when high ($wtc > 1$) they are selfless; in-between ($wtc = 1$) agents are neutral. Agents also use a *discount factor* (see below) which reflects their capacity to value short- vs long-term rewards.

The agents move synchronously around the grid world, being capable of five actions: up, down, left, right and work (remain). To simplify the analysis below, the grid is formed into a torus so that an agent moving up from the top of the grid arrives at the bottom in the corresponding column; similarly for the left and right edges. This enables us to use a relatively simple probabilistic model of square occupancy by agents.

This scenario offers a broad range of environments in which the effectiveness of reasoning about the various CMs can be assessed. While this scenario is obviously idealised (in the tradition of the tileworld scenario for single agent reasoning), we believe it incorporates the key facets that an agent would face when making coordination decisions in realistic environments. In particular, the ability to systematically control variability in the scenario is needed to evaluate our claims about the efficacy of flexible coordination and the dynamic selection of CMs according to prevailing circumstances.

4 Agent Decision-Making

Since the agents always have a specific task to achieve with $mpr = 1$ and $effort = 1$ they always have a non-zero expectation of their future reward. However, they can increase this reward by initiating or participating in collaborative tasks (CTs) with other agents. In deciding on its collaborations, each agent aims to maximise its future rewards by adopting the best mechanism under the circumstances. Thus at all instants in time, an agent will have a current goal being undertaken using a particular CM with the agent taking a particular role.

Agents may find tasks in the environment or be offered them by other agents. In either case, however, an agent needs a means of evaluating new tasks under each of the CMs. It must also be able to compare these values with its current task so as to determine when to accept a new offer. Thus each CM has an associated evaluation function which it can use to approximate the value of potential new tasks, as well as any current task being undertaken using it. These evaluation functions have some common features, but they also differ between the CMs in that they may use different aspects of the (perceived) environment.

An important consideration that needs to be incorporated into the evaluation of new offers is that of liability. If, for example, an agent has agreed to participate under the Contract Net CM, it may incur a penalty for decommitting – any new offer will need to be greater than the current task *and* the decommitment penalty. Conversely, a manager must manage its liabilities so that, if a task becomes unprofitable, it can "cut its losses" and abandon it (since the situation has already deteriorated from the time it was adopted).

Our agents are myopic in that they can only see as far as the next reward, however since tasks may arrive at different times in the future we discount all rewards back to the present using a discount factor, $0 < \delta < 1$. When $\delta \approx 1$, the difference between long- and short-term rewards is not great; however, when $\delta << 1$, short-term rewards appear more attractive [7].

For the Social Law, Pot Luck and Contract Net CMs both managers and subs mask their valuations according to their own wtc factor. This makes coordinating over collaborative tasks a more attractive proposition when wtc is high and less attractive when it is low. Evaluation of the Asocial CM is unaffected by wtc.

The following subsections give details of the evaluation functions used for each of the aforementioned CMs. These functions are designed to illustrate the sorts of functions that can be used to evaluate CMs. They are necessarily *approximate* valuations, not least because there is a great deal of uncertainty and imperfect information in the scenario. We do not claim that they are the only ones possible, nor that they are optimal, but rather that they are reasonable and demonstrate that CMs can be evaluated in dynamic and unpredictable environments using suitably parameterised functions.

4.1 Asocial CM

This CM simply involves the agent moving towards its task and working there for the necessary number of time steps before receiving the reward. To evaluate a task with $mpr = 1$, $effort = e$ and $reward = R$, an agent discounts the reward it expects by the time until it will receive it. If the distance to the task is l, the expected value of the CM, V_A, is given by:

$$V_A = R\delta^{l+e}$$

If the agent is already performing the task ($l = 0$), the reward is discounted by the amount of effort remaining.

Evaluation of the Asocial CM does not require any additional environmental information.

4.2 Social Law CM

When an agent adopts the Social Law CM, the appropriate number of the nearest agents are commanded to come to the manager's assistance. When the final agent arrives, all agents work on the task until completion and the reward is equally divided among them.

To evaluate a task with $mpr = m$, $effort = e$ and $reward = R$ under this CM, the agent must estimate the time it will take for the furthest agent to arrive at the square. This can be calculated using the average occupancy of each square[2]. The manager adds up the occupancies of the nearest squares to it until it obtains $m - 1$, and uses the furthest square required to estimate the time till all agents will be present, say l. Since this CM can only be invoked on one task at a time, the manager may also take some non-zero set up cost into account, say s. Given these estimates, the expected value of this CM, V_{SL}, is given by:

$$V_{SL} = \frac{R\delta^{(l+s+\frac{e}{m})}}{m}$$

Evaluation of the Social Law CM requires knowledge of the distribution and density of other agents, here represented by average occupancy, as well as any set up costs which using the social law may incur.

4.3 Pot Luck CM

When an agent adopts the Pot Luck CM, it makes no effort to recruit other agents unless they happen to enter the square where the CT is situated. In such cases, the manager offers the potential subordinates employment for an indefinite period at a fixed rate. The terms it offers reflect the agent's perceptions about the *wtc* and discount factor of other agents; it sets a rate that, it believes, is sufficient to attract passers-by until the task has been achieved. Any agents that accept this offer and remain at the square, committed to this task, receive the agreed rate at each step. These offers of piecemeal employment remain until the task is completed or the managing agent abandons it. When the task is complete, the managing agent receives the reward.

[2] That is, $\frac{numberOfOtherAgents}{numberOfSquares}$.

To evaluate a task with $mpr = m$, $effort = e$ and $reward = R$, the manager assumes that subs will wander by at intervals determined by the remaining average occupancy: if n agents are at large, the interval is given by *numberOfSquares* $/n$. The manager computes a rate, r, to offer to the subs that it would find acceptable itself in similar circumstances (see below). Based on these assumptions and the task *effort*, the agent computes the expected completion time of the task, *ect* and the future value of the amount it will have to pay out to the subs, p. Then the expected value of applying this CM, V_{PL}, is given by:

$$V_{PL} = (R - p)\delta^{ect}$$

When this CM is in use, the manager uses a similar technique to evaluate the task in progress, taking into account any subs already helping. Clearly, if agents are already present, the value increases considerably. Note that, with the Pot Luck CM, it is possible that the manager recruits more agents than the *mpr*, meaning that the task will be achieved more quickly.

Subordinates evaluating a Pot Luck offer discount the rate offered, r, indefinitely into the future:

$$V_{PL} = \frac{r}{1 - \delta}$$

Thus although the rate may be low compared with the reward for their default task, the fact that they will receive it regularly starting from the next time step makes the offer relatively attractive.

Evaluation of the Pot Luck CM again requires knowledge of the distribution and density of other agents, though this knowledge is used in a different way. Managers need to be able to offer an appropriate rate to subs; in general, the manager will need to consider the other agents' *wtc* and discount factors, though here it assumes them to be the same as its own.

4.4 Contract Net CM

Under this mechanism, a manager broadcasts a request for bids to the other agents. On receiving their replies, it computes the best set of bids and sends out firm offers of employment to the selected agents. These agents may either accept the offer or decline, possibly causing the managing agent to issue more requests. When the task has been completed, the manager receives the reward and pays its recruits the agreed amounts. Since the completion time of the task is unknown, subordinates bid an amount which reflects their current requirements, and when they are paid off this is factored up by their discount rate and their time committed. This means that the manager has an increasing liability to the other agents so long as the task remains unfinished—any extension to the *ect* may therefore greatly affect the value of using this CM.

To evaluate this mechanism, the agent estimates the average distance away of the furthest agent (as described above) and adds to this the communication costs of this CM (3 time steps until subs will be committed) and the duration based on *effort*/*mpr*. This gives the *ect*. The manager also estimates the likely bid requirements of the subs based on its perception of their *wtc*, discount factors and their specific task rewards. Thus if it anticipates completion in *ect* time steps, with i subs committed 3 time steps hence each bidding r_i, the value of using the Contract Net CM is given by:

$$V_{CN} = \delta^{ect}(R - \sum_i \frac{r_i}{\delta_i^{ect-3}})$$

The reward structure of this CM is such that the subs only receive payment when the manager either completes or abandons the task. At this time they receive their bid factored up by the amount of time they have been committed (agent i receives r_i/δ_i^t after being committed for t time steps). Thus, however long they are committed to the task, the reward they receive, discounted back to when they started, remains the same. In this way the manager can determine its current liability at any stage of the coordination, e.g., when an offer is rejected.

An agent bidding under the Contract Net CM, factors its current reward by its *wtc* and projects it two time steps into the future, since this is when an offer will arrive. It submits this amount, r_i, plus its required discount factor, δ_i, and the time it expects to arrive at the task. The manager selects bids based on how they affect its expected reward. That is, it looks at the surplus reward when each agent has been paid, discounted by how long until it arrives. This simplifies the otherwise combinatorial problem of selecting the best i bids.

Evaluating a Contract Net CM that is underway, involves computing the current liability to agents committed, projecting this forward till the *ect* and discounting this value and the reward back to the present. As time goes by, the value of participating in this CM increases for subs (since the manager will be paying more and more at each time step) and so a sub becomes more committed the longer it has been involved.

If a sub decommits, even through being coopted under Social Law, it must pay the manager a decommitment penalty, which is intended to compensate the manager for the time wasted. Although not implemented here, a more sophisticated evaluation function would require an estimate of the likelihood of subs decommitting. Additionally, more flexible decommitment penalties could be set dynamically to reflect the prevailing circumstances. This aspect of our work is being investigated concurrently and is reported in [6].

Evaluation of the Contract Net CM requires similar knowledge to the Pot Luck CM.

5 Experiments, Results and Analysis

Our first set of experiments assessed the accuracy of each CM's evaluation function; the circumstances under which each is selected; and to what extent this additional flexibility benefits the managing agent. To this end, we conducted a series of simulations for a 10 x 10 grid, containing just one CT with *reward* = 15 (default tasks have *reward* = 1). Furthermore, all agents have *wtc* = 1 and *discount* = 0.9. We varied the number of agents (from 5 to 25), the *mpr* (from

1 to 8) and the *effort* (from 10 to 30). Since this represents a very large sample space, we report a selection of representative results.

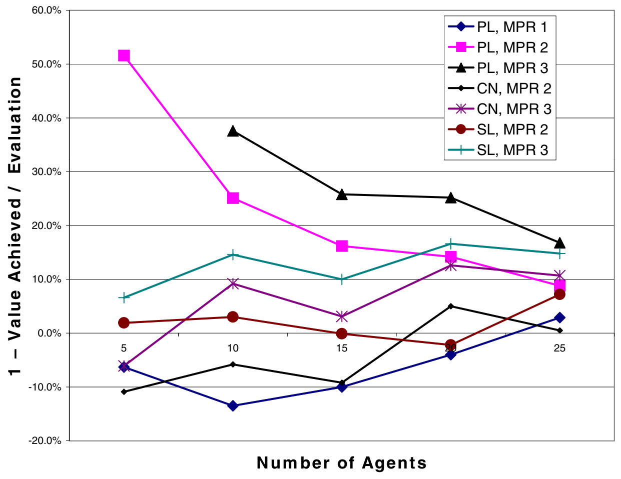


Fig. 1. Performance of CM Evaluation Functions

Figure 1 plots the difference between the value managers actually achieved and their evaluations, for each CM, $mpr = 1, 2, 3$ and $\mathit{effort} = 10$. (SL and CN do not apply when $mpr = 1$; the value of PL when $mpr = 3$ and there are only 5 agents is negative and so the manager sticks with its default task.) PL is the least accurate, overestimating the true value by 50% in the worst case. SL and CN are more accurate reflecting their lower levels of uncertainty. The graph indicates that the accuracy of all CMs tends to improve as the agent population increases. Bearing in mind that, for a discount factor of 0.9, a one step error in *ect* leads to about a 10% error in evaluation, these results show that the evaluation functions work acceptably under a variety of conditions. This demonstrates that it is feasible to evaluate CMs based on estimates of environmental parameters.

Figure 2 shows the type of CM selected by the managers under different agent densities and task profiles. It is clear that, at least from the manager's point of view, for high agent densities and low *mpr*, PL is the most preferred. This is due to PL's low communication costs (no active recruitment), low payout rate to subordinates and the ability to over-recruit. However, under more general conditions, CN is the most preferred: the communication cost (set up delay) and the relatively high rates paid to subs are compensated for by the reduced time till all agents arrive. For tasks with extremely stringent requirements, i.e., high *mpr* or *effort* in low density populations, the speed and certainty of SL leads to it being selected over CN. In fact, the choice between CN and SL depends mainly on the reward assigned to the task—for lower task rewards, SL is preferable to

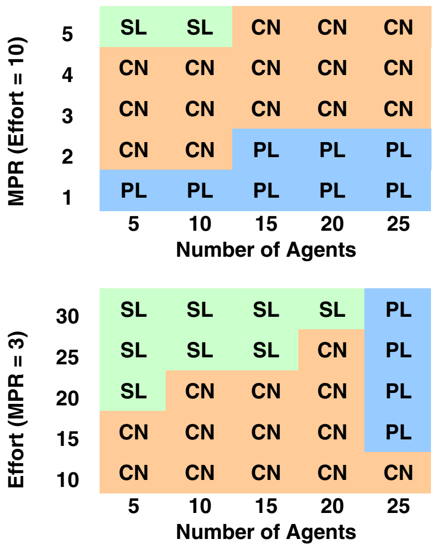


Fig. 2. Preferred CM by Task/Environment

CN. In the extreme, even SL is unprofitable and managers opt to perform their default tasks instead.

Figure 3 shows the total reward achieved by managers using each CM in isolation, and when the managers have the capability to dynamically choose which CM to use. The number of agents is fixed at 15, and the task profiles have fixed *reward* = 15, *effort* = 10 and *mpr* ranging from 1 to 8. As the *mpr* increases, the surplus reward available to managers naturally decreases, but the graph shows that agents who can coordinate flexibly maintain high levels of reward (in fact the line for ALL roughly traces the maximum over all CMs).

When taken together, these results clearly show that providing alternative mechanisms for coordination is beneficial to agents that are required to interoperate under changeable environmental or task conditions.

Our second set of experiments examined the overall effect on the system when the agents displayed different characteristics in terms of their *wtc* and *discount* factors. The general hypothesis here is that when agents are less greedy, more collaborative tasks will be achieved and that an agent's perspective on future rewards may well affect the type of CM it chooses. To test this, we conducted a series of experiments varying *wtc* from 0.25 to 3 and *discount* from 0.5 to 0.95 (the number of agents was fixed at 15, and the tasks have fixed *reward* = 15, *effort* = 10 and *mpr* = 2).

Figure 4 shows the total number of CTs achieved for each discount factor as *wtc* increases. When the discount factor was 0.8 or less, the managers always selected SL, because this minimises the *ect*. However, when the agent's *wtc* is low, it devalues its own reward for CTs to the extent that the default task is

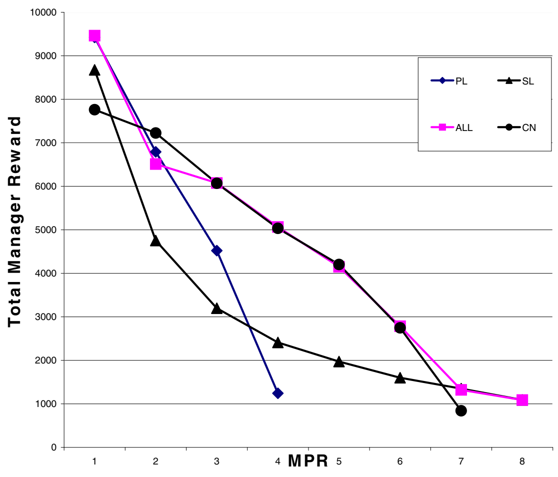


Fig. 3. Total Reward Achieved by Managers

often chosen instead. The choices for discounts of 0.9 and 0.95 were identical, with PL being chosen for $wtc \leq 1$ (agents neutral or greedy) and CN being chosen otherwise (agents selfless). This indicates that, if potential subordinates are likely to require relatively large rewards and the manager can afford to wait, it will choose to do so, paying out less to subordinates. These results confirm that the relative value of using different CMs is indeed affected by the both the individual characteristics of an agent and its beliefs about the environment in which it operates.

6 Related Work

The majority of previous work on multi-agent system coordination assumes it is a design time problem (e.g., [10,11,4]). However [5] has argued that agents need the flexibility to coordinate at different levels of abstraction, depending upon their particular needs at a given moment in time. To date, however, this work has not developed mechanisms for explicitly reasoning about which level to coordinate at in a given situation. Such flexibility was also built into cooperative problem solving agents by [9]. Here, agents could choose to cooperate according to various conventions which dictated how they should behave in a particular team context. These conventions varied in terms of the time they took to establish and the communication overhead they imposed. However, again, there was no reasoning mechanism for determining which convention was appropriate for a given situation. Boutilier [3] presents a decision making framework, based on multi-agent Markov decision processes, that does reason about the state of a coordination mechanism. However, his work is concerned with optimal reasoning within the context of a given coordination mechanism, rather than

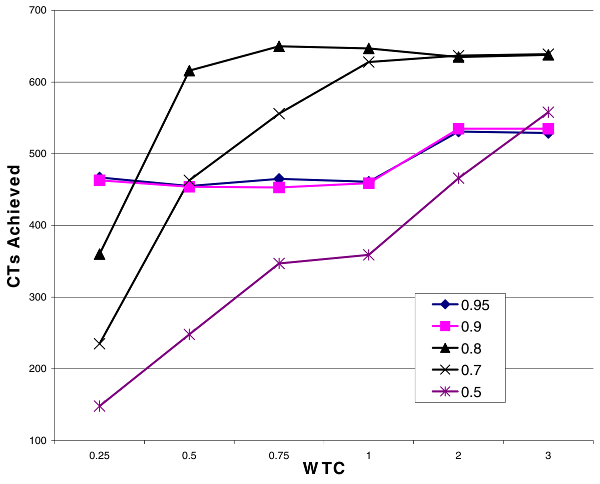


Fig. 4. CTs Achieved by WTC and Discount Factor

actually reasoning about which mechanism to employ in a particular situation. [1] present a software engineering framework that enables agents to vary their CMs according to their prevailing circumstances. They also identify criteria for determining when particular mechanisms are appropriate. However, the decision procedures for actually trading-off these criteria are not well developed. Finally, [2] provide a framework in which CMs are characterised by set up costs and probability of success and can be evaluated accordingly; however, their agents use a contract-net style protocol for all their interactions.

7 Conclusions

This paper presented a framework in which agents can evaluate and apply differing CMs and has demonstrated that agents can benefit from such flexibility. It has shown that CMs can be practically evaluated using appropriate environmental parameters despite the uncertainty agents face. However, these experiments use static environmental conditions and the agents involved use assumptions about the environment. To overcome these restrictions, in our future work we will allow agents to monitor and learn the relevant environmental parameters so that they can react to dynamic environments. Agents will then be in a position to adapt their own attributes (*wtc* and *discount*) to better suit their circumstances. Given agents that learn and adapt to their environment, it will also be important to assess whether alternative evaluation functions or heuristics impact on agent performance.

References

1. K. Barber, D. Han, and T. Liu. Coordinating distributed decision making using reusable interaction specifications. In *Proc 3rd Pacific Rim Wsp. on Multi-Agents*, pages 1–15, Australia, 2000.
2. R. A. Bourne, C. B. Excelente-Toledo, and N. R. Jennings. Run-time selection of coordination mechanisms in multi-agent systems. In *Proc. 14th European Conf. on AI*, pages 348–352, Berlin, Germany, 2000.
3. C. Boutilier. Sequential optimality and coordination in multiagent systems. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence (IJCAI-99)*, pages 478–485, Menlo Park, CA, 1999.
4. E. Durfee and V. Lesser. Partial global planning. *IEEE Trans. on Systems, Man, and Cybernetics*, 21(5):1167–1183, 1991.
5. E. H. Durfee. Practically coordinating. *AI Magazine*, 20(1):99–116, 1999.
6. C. B. Excelente-Toledo, R. A. Bourne, and N. R. Jennings. Reasoning about commitments and penalties for coordination between autonomous agents. In *Proceedings of 5th International Conference on Autonomous Agents (Agents 2001)*, pages 131–138, 2001.
7. J. Filar. *Competitive Markov decision processes*. Springer, New York, 1997.
8. A. Haddadi. *Communication and cooperation in agent systems*. Springer Verlag, 1996.
9. N. R. Jennings. Commitments and conventions. *The Knowledge Engineering Review*, 8(3):223–250, 1993.
10. Y. Shoham and M. Tennenholtz. On the synthesis of useful social laws for artificial agent societies. In *Proc. of the 10th Nat. Conf. on AI*, San Diego, CA, 1992.
11. R. G. Smith and R. Davis. Frameworks for cooperation in distributed problem solving. *IEEE Trans. on Systems, Man, and Cybernetics*, 11(1):61–70, 1980.

Evolving Multi-agent Viewpoints – An Architecture

Pierangelo Dell'Acqua[1], João Alexandre Leite[2], and Luís Moniz Pereira[2]

[1] Department of Science and Technology, Campus Norrköping
Linköping University, Norrköping, Sweden, pier@itn.liu.se
[2] Centro de Inteligência Artificial - CENTRIA, Universidade Nova de Lisboa
2829-516 Caparica, Portugal, {jleite | lmp}@di.fct.unl.pt

Abstract. We present an approach to agents that can reason, react to the environment and are able to update their own knowledge as a result of new incoming information. Each agents' view of the social relationships among agents (itself and others) is represented by a graph, which in turn can be updated, allowing for the representation of such social evolution.

1 Introduction and Motivation

Over recent years, the notion of agency has claimed a major role in defining the trends of modern research. Influencing a broad spectrum of disciplines such as Sociology, Psychology, among others, the agent paradigm virtually invaded every sub-field of Computer Science. Besides allowing for a unified declarative and procedural semantics, eliminating the traditional wide gap between theory and practice, the use of several and quite powerful results in the field of non-monotonic extensions to *Logic Programming* (LP), such as belief revision, inductive learning, argumentation, preferences, abduction, etc.[15,17] can represent an important added value to the design of rational agents. These results, together with the improvement in efficiency (cf. [5,16,18]), allow *Logic Programming* and *Non-monotonic Reasoning* to accomplish a fruitful degree of combination between reactive and rational behaviours of agents, whilst preserving clear and precise specification enjoyed by declarative languages. This goal in mind, Kowalski and Sadri [11] advanced an agent architecture based on an observe-think-act cycle. It was further developed by Dell'Acqua and Pereira [6] to allow the agents to dynamically update their own knowledge base (whether intentional or extensional) as well as their own goals. The capability of updating provided to these agents is inherited by *Dynamic Logic Programming* [1].

Dynamic Logic Programming (DLP) was introduced, following the eschewing of performing updates on a model basis, to envisage updates as applying to logic programming rules making up a theory [14]. According to DLP, knowledge is conveyed by a set of theories (encoded as generalized logic programs) representing different states of the world. Different states may represent distinct dimensions such as different time periods, different hierarchical instances, or even different domains. Consequently, the individual theories may contain mutually contradictory and overlapping information. The role of DLP is to take into account the mutual relationships extant between different states to precisely determine the declarative and the procedural semantics of the combined theory comprised of all individual theories and the way they relate.

Although DLP can represent several states in one evolving dimension or aspect of a system, no more than one such aspectual evolution can be encoded and combined simultaneously. This is so because DLP is defined only for linear sequences of states. In [15], the states represent different time instants. To overcome this drawback, Leite et al. introduced *Multi-dimensional Dynamic Logic Programming* (MDLP) [12], which generalizes DLP to allow for collections of states organized in arbitrary directed acyclic graphs (DAGs). Within this more general theory, one can encode simultaneously all representational dimensions, which can be particularly useful in the context of multi-agent systems.

In this paper, we formalize agents with such capabilities, generalizing the framework of [15] to allow for an arbitrary number of dimensions represented by an arbitrary DAG. We will show how this new theory is useful for an agent to represent and reason about its own and other agents' knowledge and its evolution in time. Each agent's view of the evolving social relationships among agents (itself and others) is represented by a DAG, itself in turn updatable, to capture the representation of social evolution.

The contribution of the paper is therefore twofold. On the one side, the paper presents an extension of the framework for multi-dimensional dynamic logic programming to incorporate integrity constraints and active rules, and to make the DAG of each agent updatable. On the other, the paper provides a means to incorporate the obtained framework into a multi-agent architecture. For simplicity, we have considered propositional generalized logic programs.

2 Preliminaries

Graphs A *directed graph*, or *digraph*, $D = (V, E)$ is a pair of two finite or infinite sets $V = V_D$ of *vertices* and $E = E_D$ of pairs of vertices or (*directed*) *edges*. A *directed edge sequence from* v_0 *to* v_n in a digraph is a sequence of edges $e_1, e_2, ..., e_n \in E_D$ such that $e_i = (v_{i-1}, v_i)$ for $i = 1, ..., n$. A *directed path* is a directed edge sequence in which all the edges are distinct. A *directed acyclic graph*, or *acyclic digraph (DAG)*, is a digraph D such that there are no directed edge sequences from v to v, for all vertices v of D. A *source* is a vertex with in-valency 0 (number of edges for which it is a final vertex) and a *sink* is a vertex with out-valency 0 (number of edges for which it is an initial vertex). We say that $v < w$ if there is a directed path from v to w and that $v \leq w$ if $v < w$ or $v = w$. The *transitive closure* of a graph D is a graph $D^+ = (V, E^+)$ such that for all $v, w \in V$ there is an edge (v, w) in E^+ if and only if $v < w$ in D. The relevancy DAG of a DAG D wrt a vertex v of D is $D_v = (V_v, E_v)$ where $V_v = \{v_i : v_i \in V \text{ and } v_i \leq v\}$ and $E_v = \{(v_i, v_j) : (v_i, v_j) \in E \text{ and } v_i, v_j \in V_v\}$.

Logic Programming Framework In order to represent negative information in logic programs, we need more general logic programs that allow default negation *not* A not only in premises of their clauses but also in their heads[1]. We call such programs generalized logic programs. It is convenient to syntactically represent generalized logic programs as propositional Horn theories. In particular, we represent default negation *not* A as a standard propositional variable.

Propositional variables whose names do not begin with "*not*" and do not contain the symbols ":" and "÷" are called **objective atoms**. Propositional variables of the form *not* A are called **default atoms**. Objective atoms and default atoms are generically called **atoms**.

Propositional variables of the form $i{:}C$ (where C is defined below) are called **projects**. $i{:}C$ denotes the intention (of some agent j) of proposing the updating the theory of agent i with C. Propositional variables of the form $j \div C$ are called **updates**. $j \div C$ denotes an update that has been proposed by j of the current theory of agent (of some agent i) with C. We assume that updates cannot be negated (i.e., we disallow *not* $i{\div}C$). Instead projects can be negated. A negated project *not* $i{:}C$ denotes the intention of the agent not to perform project $i{:}C$. Let $\mathcal{K}$ be a set of propositional variables consisting of objective atoms and projects such that $\mathit{false} \notin \mathcal{K}$. The propositional language $\mathcal{L}_{\mathcal{K}}$, generated by $\mathcal{K}$, consists of the following set of propositional variables: $\mathcal{L}_{\mathcal{K}} = \mathcal{K} \cup \{\mathit{not}\, A \mid \forall \text{ atom } A \in \mathcal{K}\} \cup \{i{\div}C, \mathit{not}\, i{:}C \mid \forall i{:}C \in \mathcal{K}\}$. A **generalized rule** in $\mathcal{L}_{\mathcal{K}}$ is of the form: $L_0 \leftarrow L_1 \wedge \ldots \wedge L_n,\ (n \geq 0)$ where every L_i $(0 \leq i \leq n)$ is an atom from $\mathcal{L}_{\mathcal{K}}$. An **integrity constraint** in $\mathcal{L}_{\mathcal{K}}$ is a rule of the form: $\mathit{false} \leftarrow L_1 \wedge \ldots \wedge L_n \wedge Z_1 \wedge \ldots \wedge Z_m,\ (n \geq 0, m \geq 0)$ where every L_i $(1 \leq i \leq n)$ is an atom, and every Z_j $(1 \leq j \leq m)$ is a project from $\mathcal{L}_{\mathcal{K}}$. Integrity constraints are rules that enforce some condition over the state, and therefore take the form of denials. A **generalized logic program** P in $\mathcal{L}_{\mathcal{K}}$ is a set of generalized rules and integrity constraints in $\mathcal{L}_{\mathcal{K}}$. A **query** Q in $\mathcal{L}_{\mathcal{K}}$ is of the form: $?{-}\ L_1 \wedge \ldots \wedge L_n,\ (n \geq 1)$ where every L_i $(1 \leq i \leq n)$ is an atom from $\mathcal{L}_{\mathcal{K}}$. The following definition introduces rules that are evaluated bottom-up. To emphasize this aspect, we employ a different notation for them. An **active rule** $\mathcal{L}_{\mathcal{K}}$ is a rule of the form: $L_1 \wedge \ldots \wedge L_n \Rightarrow Z,\ (n \geq 0)$ where every L_i $(1 \leq i \leq n)$ is an atoms, and Z is a project from $\mathcal{L}_{\mathcal{K}}$. Active rules are rules that modify the current state when executed. Active rules take the form: $\mathit{action\_name} \wedge \mathit{Preconditions} \Rightarrow Z$ where *action_name* is an abducible. If the *Preconditions* of the rule are satisfied, then the project (fluent) Z can be selected and executed. The head of an active rule must be a project that is either internal or external. An *internal project* operates on the state of the agent, e.g., if an agent gets an observation, then it updates its knowledge, or if some conditions are met, then it proves some goal, etc. *External projects* instead are performed on the environment, e.g., when an agent sends a message to another agent.

Given a set of vertices V, we assume that for every project $i{:}C$ in $\mathcal{K}$, C is either a generalized rule, an integrity constraint, an active rule, a query or an atom of the form $\mathit{modify\_edge}(j,u,v,x)$, $\mathit{not}\ \mathit{modify\_edge}(j,u,v,x)$, $\mathit{not}\ \ \mathit{edge}(u,v)$, or $\mathit{edge}(u,v)$ where $u,v \in V$. Thus, a project can only take one of the following

[1] For further motivation and intuitive reading of logic programs with default negations in the heads see [1].

form:

$$\begin{array}{ll} i{:}(L_0 \leftarrow L_1 \wedge \ldots \wedge L_n) & i{:}(L_1 \wedge \ldots \wedge L_n \Rightarrow Z) \\ i{:}(\mathit{false} \leftarrow L_1 \wedge \ldots \wedge L_n \wedge Z_1 \wedge \ldots \wedge Z_m) & i{:}(?{-}L_1 \wedge \ldots \wedge L_n) \\ i{:}\mathit{modify\_edge}(j,u,v,x) & i{:}\mathit{edge}(u,v) \\ i{:}\mathit{not\ modify\_edge}(j,u,v,x) & i{:}\ \mathit{not\ edge}(u,v) \end{array}$$

Note that the predicates *edge* and *modify_edge* can only occur inside projects or updates since those predicates do not belong to $\mathcal{L}_\mathcal{K}$. We assume that $i{:}\mathit{edge}(u,v)$ and $i{:}(?{-}L_1,\ldots,L_n)$ can only be internal projects, that is, only the agent itself can issue a project to update its own DAG (i.e., $\mathit{edge}(u,v)$) and goals (i.e., $?{-}L_1,\ldots,L_n$). The project $i{:}\mathit{edge}(u,v)$ issued by the agent i denotes the intention of i to modify its own DAG by establishing an edge between the vertices u and v in V. By issuing a project $i{:}\mathit{modify\_edge}(j,u,v,x)$ the agent i expresses the intention to modify the DAG of another agent j by adding/deleting (depending on whether $x = a$ or $x = d$) an edge between u and v.

3 Agents and Multi-agent Systems

This section presents the notion of agent and multi-agent system. The initial theory of an agent is characterized by a multi-dimensional abductive logic program, inspired by [12], which expresses the agent's viewpoint on the relationships amongst a collection of agents, and encoded in a directed acyclic graph.

Definition 1. Let $\mathcal{L}_\mathcal{K}$ be a propositional language. Let $\mathcal{M}$ be a multi-agent system (defined below) and α an agent in $\mathcal{M}$. A **multi-dimensional abductive logic program** for α, written as Ψ_α, is a tuple $\mathcal{T} = \langle D, \mathcal{P}_D, \mathcal{A}, \mathcal{R}_D \rangle$ where:

- $D = (V, E)$ is an acyclic directed graph, $V = \{v \mid \text{for every } \Psi_v \in \mathcal{M}\} \cup \{\alpha'\}$;
- $\mathcal{P}_D = \{P_v \mid v \in V\}$ is a set of generalized logic programs in the language $\mathcal{L}_\mathcal{K}$, indexed by the vertices $v \in V$ of D;
- $\mathcal{A}$ is a set of atoms;
- $\mathcal{R}_D = \{R_v \mid v \in V\}$ is a set of sets of active rules in the language $\mathcal{L}_\mathcal{K}$, indexed by the vertices $v \in V$ of D.

We call the distinguished vertex $\alpha' \in V$ the **inspection point** of agent α, and we call the atoms in $\mathcal{A}$ **abducibles**.

The initial theory of an agent α is determined (1) by a DAG D that represents the relation between the agents (represented by the vertices in V) in the multi-agent system; (2) by a set of generalized logic programs, one program P_v for each agent $v \in V$; (3) by a set of abducibles; and (4) by a set of sets of active rules R_v, for each agent $v \in V$.

Without loss of generality, we can assume that abducibles have no matching clauses in $P = \bigcup_{v \in V} P_v$. Abducibles can be thought of as hypotheses that can be used to extend the given logic program in order to provide an "explanation", or conditional answer, to a query. Explanations are required to satisfy the integrity constraints in P.

The multi-dimensional abductive logic program formalizes the initial knowledge state of the agent and its reactive behaviour. The knowledge of an agent can dynamically evolve when the agent receives new knowledge, albeit by self-updating rules. This is represented in the form of an updating program.

Definition 2. Let $\mathcal{M}$ be a multi-agent system (defined below). An **updating program** U is a finite set of updates such that if $v \div C \in U$ then $\Psi_v \in \mathcal{M}$.

An updating program contains the updates that will be performed on the current knowledge state of the agent. To characterize the evolution of the knowledge of an agent we need to introduce the notion of sequence of updating programs. Let $S = \{0, 1, \ldots, m\}$ be a set of natural numbers. We call the elements $s \in S$ *states*. A *sequence of updating programs* $\mathcal{U} = \{U^s \mid s \in S \text{ and } s > 0\}$ is a set of updating programs U^s superscripted by the states $s \in S$.

Definition 3. Let $s \in S$ be a state. An **agent** α **at state** s, written as Ψ_α^s, is a pair $(\mathcal{T}, \mathcal{U})$, where $\mathcal{T}$ is the multi-dimensional abductive logic program of α, and $\mathcal{U} = \{U^1, \ldots, U^s\}$ is a sequence of updating programs. If $s = 0$, then $\mathcal{U} = \{\}$.

An agent α at state 0 is defined by its initial theory and an empty sequence of updating programs, that is $\Psi_\alpha^0 = (\mathcal{T}, \{\})$. At state 1, α is defined by $(\mathcal{T}, \{U^1\})$, where U^1 is the updating program containing all the updates that α has received at state 1, either from other agents or as self-updates. In general, an agent α at state s is defined by $\Psi_\alpha^s = (\mathcal{T}, \{U^1, \ldots, U^s\})$, where each U^i is the updating program containing the updates that α has received at state i.

Definition 4. A **multi-agent system** $\mathcal{M} = \{\Psi_{v_1}^s, \ldots, \Psi_{v_n}^s\}$ at state s is a set of agents $v_1, \ldots, v_n$ each of which at state s.

Note that the definition of multi-agent system characterizes a static society of agents in the sense that it is not possible to add/remove agents from the system, and all the agents are at one common state.

To begin with, the system starts at state 0 where each agent is defined by $\Psi_\alpha^0 = (\mathcal{T}_\alpha, \{\})$. Suppose that at state 0 an agent β wants to propose an update of the knowledge state of agent α with C by triggering the project $\alpha : C$. Then, at the next common state α will receive the update $\beta \div C$ indicating that an update has been proposed by β. Thus, at state 1, α will be $(\mathcal{T}_\alpha, \{U^1\})$, where $U^1 = \{\beta \div C\}$ if no other updates occur for α.

Within logic programs we refer to agents by using the corresponding subscript. For instance, if we want to express the update of the theory of an agent Ψ_α with C, we write the project $\alpha{:}C$.

The agent's semantics is given by a syntactical transformation (defined in Sect. 3.2), producing a generalized logic program for each agent, and apply to queries for it as well. Nevertheless, to increase readability, we will immediately present a case study example that, though simple, allows one to glean some of the capabilities of the framework. To increase readability, we rely on natural language when explaining some details. The agent cycle, mentioned in the example and defined in Sect. 4, operates on the results of the transformation, but the example can be understood, on a first reading, without requiring detailed knowledge of the transformations.

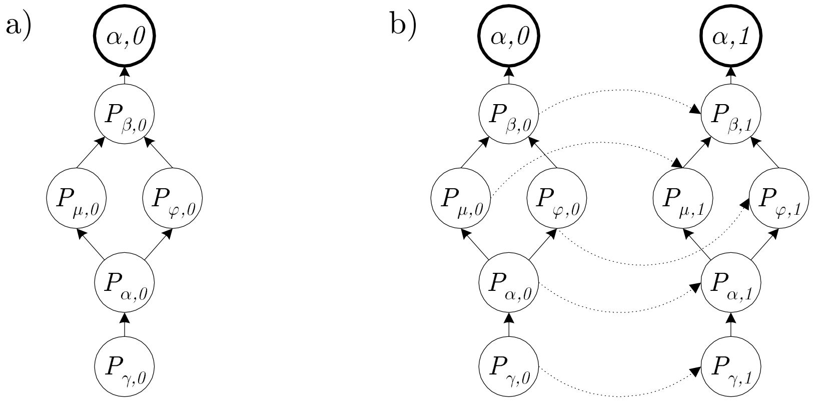


Fig. 1. *DAG of Alfredo at states 0 and 1*

3.1 Case Study

This example shows how the DAG of an agent can be modified through updates, and how its knowledge state and reactions are affected by such modifications. With this aim, we hypothesize two alternative scenarios. Alfredo lives together with his parents, and he has a girlfriend. Being a conservative lad, Alfredo never does anything that contradicts those he considers to be higher authorities, in this case his mother, his father, and the judge who will appear later in the story. Furthermore, he considers the judge to be hierarchically superior to each of his parents. As for the relationship with his girlfriend, he hears her opinions but his own always prevail. Therefore Alfredo's view of the relationships between himself and other agents can be represented by the following DAG $D = (V, E)$, depicted in Fig.1a) (where the inspection point of Alfredo, i.e. the vertex corresponding to Alfredo's overall semantics, is represented by a bold circle): $V = \{\alpha, \beta, \gamma, \mu, \varphi, \alpha'\}$ and $E = \{(\gamma, \alpha), (\alpha, \mu), (\alpha, \varphi), (\mu, \beta), (\varphi, \beta), (\beta, \alpha')\}$ where α is Alfredo, β is the judge, γ is the girlfriend, μ is the mother, φ is the father and α' is the inspection point of Alfredo.

Initially, Alfredo's programs P_{α_0} and R_{α_0} at state 0 contain the rules:

$$\begin{array}{ll}
r1 : girlfriend \leftarrow & r3 : get\_married \wedge girlfriend \Rightarrow \gamma : propose \\
r4 : not\ happy \Rightarrow \varphi : (?{-}happy) & r2 : move\_out \Rightarrow \alpha : rent\_apartment \\
r5 : not\ happy \Rightarrow \mu : (?{-}happy) & r6 : modify\_edge(\beta, u, v, a) \Rightarrow \alpha : edge(u, v)
\end{array}$$

stating that: Alfredo has a girlfriend (r1); if he decides to move out, he has to rent an apartment (r2); if he decides to get married, provided he has a girlfriend, he has to propose (r3); if he is not happy he will ask his parents how to be happy (r4, r5); Alfredo must create a new edge between two agents (represented by u and v) in his DAG every time he proves the judge told him so (r6). Alfredo's set of abducibles, corresponding to the actions he can decide to perform to reach his goals, is: $A = \{move\_out, get\_married\}$. In the first scenario, Alfredo's goal is to be happy, this being represented at the agent cycle level (cf. Sect. 4) together with *not false* (to preclude the possibility of violation of integrity constraints) and with the conjunction of active rules: $G = ?\!-(happy \wedge not\,false \wedge r2 \wedge r3 \wedge r4 \wedge r5 \wedge r6)$ During the first cycle (we will assume that there are always enough computational units to complete the proof procedure), the projects $\varphi : (? - happy)$ and $\mu : (? - happy)$ are selected. Note that these correspond to the only two active rules (r4 and r5) whose premises are verified. Both projects are selected to be performed, producing 2 messages to agents φ and μ (the reader is referred to [8] for details on how communication can be combined into this framework).

In response to Alfredo's request to his parents, Alfredo's mother, as most latin mothers, tells him that if he wants to be happy he should move out, and that he should never move out without getting married. This correspond to the observed updates $\mu \div (happy \leftarrow move\_out)$ and $\mu \div (false \leftarrow move\_out \wedge not\; get\_married)$ which produce the program P_{μ_1} at state 1 containing the following rules:

$$r7 : happy \leftarrow move\_out \qquad r8 : false \leftarrow move\_out \wedge not\; get\_married$$

Alfredo's father, on the other hand, not very happy with his own marriage and with the fact that he had never lived alone, tells Alfredo that, to be happy he should move out and live by himself, i.e. he will not be happy if he gets married now. This corresponds to the observed updates $\varphi \div (happy \leftarrow move\_out)$ and $\varphi \div (not\, happy \leftarrow get\_married)$ which produce the program P_{φ_1} at state 1 containing the following rules:

$$r9 : happy \leftarrow move\_out \qquad r10 : not\, happy \leftarrow get\_married$$

The situation after these updates is represented in Fig.1b).

After another cycle, the IFF proof procedure returns no projects because the goal is not reachable without producing a contradiction. From r7 and r8 one could abduce *move_out* to prove *happy* but, in order to satisfy r8, *get_married* would have to be abduced also, producing a contradiction via r10. This is indeed so because, according to the DAG, the rules of P_{φ_1} and P_{μ_1} cannot reject one another other since there is no path from one to the other. Thus, in this scenario, Alfredo is not reactive at all being not able to take any decision.

Consider now this second scenario where the initial theory of Alfredo is the same as in the first, but his parents have decided to get divorced. Suppose that at state 1 Alfredo receives from the judge the update $\beta \div modify\_edge(\beta, \varphi, \mu, a)$ corresponding to the decision to give Alfredo's custody to his mother after his parents divorce. This produces the program P_{β_1} containing the rule:

$$r11 : modify\_edge(\beta, \varphi, \mu, a) \leftarrow$$

After this update, the projects $\varphi : (? - happy)$, $\mu : (? - happy)$ and $\alpha : edge(\varphi, \mu)$ are selected for execution, which correspond to the evaluation of the active rules r4, r5 and r6. In response to Alfredo's request to his parents, given the precedence of opinion imposed by the judge, suppose that they reply the same rules as in the first scenario, producing the following two programs: P_{μ_2} containing the rules received from the mother:

$$r12 : happy \leftarrow move\_out \qquad r13 : false \leftarrow move\_out \wedge not\; get\_married$$

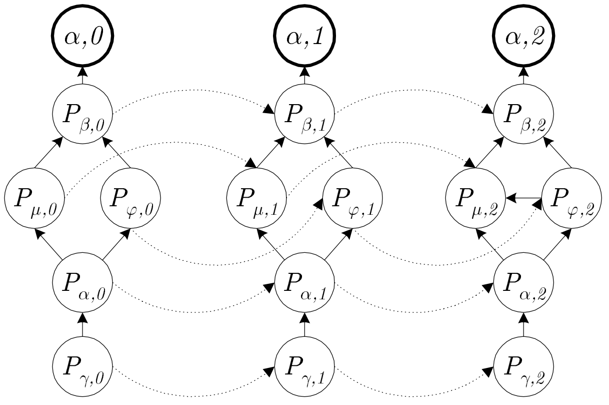


Fig. 2. *DAG of Alfredo at state 2*

and P_{φ_2} containing the rules received from the father:

$$r14 : happy \leftarrow move\_out \qquad r15 : not\, happy \leftarrow get\_married$$

The update $\alpha \div edge(\varphi, \mu)$ produces a change in the graph, and the current situation is the one depicted in Fig.2. Now, the proof procedure returns the projects α : *rent_apartment* and γ : *propose*. These projects correspond to the evaluation of rules r2 and r3 after the abduction of $\{move\_out, get\_married\}$ to prove the goal $(?{-}happy)$. Note that now it is possible to achieve this goal without reaching a contradictory state since henceforth the advice from Alfredo's mother (r12 and r13) prevails over and rejects that of his father (r14 and r15). To this second scenario, Alfredo gets married, rents an apartment, moves out with his new wife and lives happily ever after.

3.2 Syntactical Transformation

The semantics of an agent α at state m, Ψ_α^m, is established by a syntactical transformation (called *multi-dimensional dynamic program update* $\bigoplus$) that, given $\Psi_\alpha^m = (\mathcal{T}, \mathcal{U})$, produces an *abductive logic program* [6] $\langle P, \mathcal{A}, R\rangle$, where P is a normal logic program (that is, default atoms can only occur in bodies of rules), $\mathcal{A}$ is a set of abducibles and R is a set of active rules. Default negation can then be removed from the bodies of rules in P via the *dual transformation* defined by Alferes et al [3]. The transformed program P' is a definite logic program. Consequently, a version of the IFF proof procedure proposed by Kowalski et al [10] and simplified by Dell'Acqua et al [6] to take into consideration propositional definite logic programs, can then be applied to $\langle P', \mathcal{A}, R\rangle$.

In the following, to keep notation simple we write rules as $A \leftarrow B \wedge not\, C$, rather than as $A \leftarrow B1 \wedge \ldots \wedge Bm \wedge not\, C1 \wedge \ldots \wedge not\, Cn$.

Suppose that $\langle D, \mathcal{P}_D, \mathcal{A}, \mathcal{R}_D\rangle$, with $D = (V, E)$, is the multi-dimensional abductive logic program of agent α, and $S = \{1, \ldots, m\}$ a set of states. Let the vertex $\alpha' \in V$ be the inspection point of α. We will extend the DAG D with two source vertices: an initial vertex sa for atoms and an initial vertex sp for projects. We will then extend D with a set of directed edges (sa_0, s') and (sp_s, s') connecting sa to all the sources s' in D at state 0, and connecting sp to all the sources s' in D at every state $s \in S$. Let $\mathcal{K}$ be an arbitrary set of propositional variables as described above, and $\overline{\mathcal{K}}$ the propositional language extending $\mathcal{K}$ to:

$$\overline{\mathcal{K}} = \mathcal{K} \cup \left\{ \begin{array}{l} A^-, A_{v_s}, A^-_{v_s}, A_{P_{v_s}}, A^-_{P_{v_s}}, reject(A_{v_s}), \\ reject(A^-_{v_s}) \mid A \in \mathcal{K},\ v \in V \cup \{sa, sp\},\ s \in S \end{array} \right\}$$
$$\cup \left\{ \begin{array}{l} rel\_edge(u_s, v_s), rel\_path(u_s, v_s), rel\_vertex(u_s), \\ edge(u_s, v_s), edge(u_s, v_s)^- \mid u, v \in V \cup \{sa, sp\},\ s \in S \end{array} \right\}$$

Transformation of Rules and Updates

(RPR) Rewritten program rules: Let γ be a function defined as follows:

$$\gamma(v,s,F) = \begin{cases} A_{P_{v_s}} \leftarrow B \wedge C^- & \text{if } F \text{ is } A \leftarrow B \wedge not\, C \\ A^-_{P_{v_s}} \leftarrow B \wedge C^- & \text{if } F \text{ is } not\, A \leftarrow B \wedge not\, C \\ false_{P_{v_s}} \leftarrow B \wedge C^- \wedge & \text{if } F \text{ is } false \leftarrow B \wedge not\, C \wedge \\ \qquad \wedge Z1 \wedge Z2^- & \qquad \wedge Z1 \wedge not\, Z2 \\ B \wedge C^- \Rightarrow Z_{P_{v_s}} & \text{if } F \text{ is } B \wedge not\, C \Rightarrow Z \\ B \wedge C^- \Rightarrow Z^-_{P_{v_s}} & \text{if } F \text{ is } B \wedge not\, C \Rightarrow not\ Z \end{cases}$$

The rewritten rules are obtained from the original ones by replacing their heads A (respectively, the *not* A) by the atoms $A_{P_{v_s}}$ (respectively, $A^-_{P_{v_s}}$) and by replacing negative premises $not\, C$ by C^-.

(RU) Rewritten updates: Let σ be a function defined as follows:

$$\sigma(s, i \div C) = \gamma(i, s, C)$$

where $i \div C$ is not one of the cases below. Note that updates of the form $i \div (?{-}L_1 \wedge \ldots \wedge L_n)$ are not treated here. They will be treated at the agent cycle level (see Section 4). The following cases characterize the DAG rewritten updates:

$$\begin{array}{l} \sigma(s, i \div edge(u,v)) = edge(u_s, v_s) \\ \sigma(s, i \div not\ edge(u,v)) = edge(u_s, v_s)^- \\ \sigma(s, i \div modify\_edge(j,u,v,x)) = modify\_edge(j,u,v,x) \\ \sigma(s, i \div not\ modify\_edge(j,u,v,x)) = modify\_edge(j,u,v,x)^- \end{array}$$

(IR) Inheritance rules:

$$A_{v_s} \leftarrow A_{u_t} \wedge not\, reject(A_{u_t}) \wedge rel\_edge(u_t, v_s)$$
$$A^-_{v_s} \leftarrow A^-_{u_t} \wedge not\, reject(A^-_{u_t}) \wedge rel\_edge(u_t, v_s)$$

for all objective atoms $A \in \mathcal{K}$, for all $u, v \in V \cup \{sa\}$ and for all $s, t \in S$. The inheritance rules state that an atom A is true (resp., false) in a vertex v at state s if it is true (resp., false) in any ancestor vertex u at state t and it is not rejected, i.e., forced to be false.

(RR) Rejection Rules:

$$reject(A^-_{u_s}) \leftarrow A_{P_{v_t}} \wedge rel\_path(u_s, v_t) \qquad reject(A_{u_s}) \leftarrow A^-_{P_{v_t}} \wedge rel\_path(u_s, v_t)$$

for all objective atoms $A \in \mathcal{K}$, for all $u, v \in V \cup \{sa\}$ and for all $s, t \in S$. The rejection rules say that if an atom A is true (respectively, false) in the program P_v at state t, then it rejects inheritance of any false (respectively, true) atoms of any ancestor u at any state s.

(UR) Update Rules:

$$A_{v_s} \leftarrow A_{P_{v_s}} \qquad A^-_{v_s} \leftarrow A^-_{P_{v_s}}$$

for all objective atoms $A \in \mathcal{K}$, for all $v \in V \cup \{sa\}$ and for all $s \in S$. The update rules state that an atom A must be true (respectively, false) in the vertex v at state s if it is true (respectively, false) in the program P_v at state s.

(DR) Default Rules:

$$A^-_{sa_0}$$

for all objective atoms $A \in \mathcal{K}$. Default rules describe the initial vertex sa for atoms (at state 0) by making all objective atoms initially false.

(CSR$_{v_s}$) Current State Rules:

$$A \leftarrow A_{v_s} \qquad A^- \leftarrow A^-_{v_s}$$

for every objective atoms $A \in \mathcal{K}$. Current state rules specify the current vertex v and state s in which the program is being evaluated and determine the values of the atoms A and A^-.

Transformation of projects

(PPR) Project Propagation rules:

$$Z_{u_s} \wedge not\, rejectP(Z_{u_s}) \wedge rel\_edge(u_s, v_s) \Rightarrow Z_{v_s}$$
$$Z^-_{u_s} \wedge not\, rejectP(Z^-_{u_s}) \wedge rel\_edge(u_s, v_s) \Rightarrow Z^-_{v_s}$$

for all projects $Z \in \mathcal{K}$, for all $u, v \in V \cup \{sp\}$ and for all $s \in S$. The project propagation rules propagate the validity of a project Z through the DAG at a given state s. In contrast to inheritance rules, the validity of projects is not propagated along states.

(PRR) Project Rejection Rules:

$$rejectP(Z^-_{u_s}) \leftarrow Z_{P_{v_s}} \wedge rel\_path(u_s, v_s)$$
$$rejectP(Z_{u_s}) \leftarrow Z^-_{P_{v_s}} \wedge rel\_path(u_s, v_s)$$

for all projects $Z \in \mathcal{K}$, for all $u, v \in V \cup \{sp\}$ and for all $s \in S$. The project rejection rules state that if a project Z is true (respectively, false) in the program P_v at state s, then it rejects propagation of any false (respectively, true) project of any ancestor u at state s.

(PUR) Project Update Rules:

$$Z_{P_{v_s}} \Rightarrow Z_{v_s} \qquad Z^-_{P_{v_s}} \Rightarrow Z^-_{v_s}$$

for all projects $Z \in \mathcal{K}$, for all $v \in V \cup \{sp\}$ and for all $s \in S$. The project update rules declare that a project Z must be true (respectively, false) in v at state s if it is true (respectively, false) in the program P_v at state s.

(PDR) Project Default Rules:

$$Z^-_{sp_s}$$

for all projects $Z \in \mathcal{K}$ and for all $s \in S$. Project default rules describe the initial vertex *sp* for projects at every state $s \in S$ by making all projects initially false.

(PAR$_{v_s}$) Project Adoption Rules:

$$Z_{v_s} \Rightarrow Z$$

for every project $Z \in \mathcal{K}$. These rules, only applying to positive projects, specify the current vertex v and state s in which the project is being evaluated. Projects evaluated to true are selected at the agent cycle level and executed.

Transformation rules for edge

(ER) Edge Rules:

$$edge(u_0, v_0)$$

for all $(u, v) \in E$. Edge rules describe the edges of D at state 0. Plus the following rules that characterize the edges of the initial vertices for atoms (sa) and projects (sp): $edge(sa_0, u_0)$ and $edge(sp_s, u_s)$ for all sources $u \in V$ and for all $s \in S$.

(EIR) Edge Inheritance rules:

$$\begin{aligned} edge(u_{s+1}, v_{s+1}) &\leftarrow edge(u_s, v_s) \wedge not\ edge(u_{s+1}, v_{s+1})^- \\ edge(u_{s+1}, v_{s+1})^- &\leftarrow edge(u_s, v_s)^- \wedge not\ edge(u_{s+1}, v_{s+1}) \\ edge(u_s, u_{s+1}) &\leftarrow \end{aligned}$$

for all $u, v \in V$ such that u is not the inspection point of the agent, and for all $s \in S$. Note that EIR do not apply to edges containing the vertices *sa* and *sp*.

(RER$_{v_s}$) Current State Relevancy Edge Rules:

$$rel\_edge(X, v_s) \leftarrow edge(X, v_s) \quad rel\_edge(X, Y) \leftarrow edge(X, Y) \wedge rel\_path(Y, v_s)$$

Current state relevancy edge rules define which edges are in the relevancy graph wrt. the vertex v at state s.

(RPR) Relevancy Path Rules:

$$\begin{aligned} rel\_path(X, Y) &\leftarrow rel\_edge(X, Y) \\ rel\_path(X, Z) &\leftarrow rel\_edge(X, Y) \wedge rel\_path(Y, Z). \end{aligned}$$

Relevancy path rules define the notion of relevancy path in a graph.

3.3 Multi-dimensional Updates for Agents

This section introduces the notion of multi-dimensional dynamic program update, a transformation that, given an agent $\Psi^m_\alpha = (\mathcal{T}, \mathcal{U})$, produces an abductive logic program (as defined in [6]). Note first that an updating program U for an agent α can contain updates of the form $\alpha \div (?{-}L_1 \wedge \ldots \wedge L_n)$. Such updates are intended to add $L_1 \wedge \ldots \wedge L_n$ to the current query of the agent α. Thus, those updates have to be treated differently. To achieve this, we introduce a function β defined over updating programs. Let U be an updating program and α an agent in $\mathcal{M}$. If U does not contain any update starting with "$\alpha \div (?{-}$", then, $\beta(\alpha, U) = (\mathit{true}, U)$. Otherwise, suppose that U contains n updates starting with "$\alpha \div (?{-}$", say $\alpha \div (?{-}C_1), \ldots, \alpha \div (?{-}C_n)$ for n conjunctions $C_1, \ldots, C_n$ of atoms. Then, $\beta(\alpha, U) = (C_1 \wedge \ldots \wedge C_n, U - \{\alpha \div (?{-}C_1), \ldots, \alpha \div (?{-}C_n)\})$. Given a set Q of rules, integrity constraints and active rules, we write Ω_1 and Ω_2 to indicate: $\Omega_1(Q) = \{C \mid \text{for every rule and integrity constraint } C \text{ in } Q\}$ and $\Omega_2(Q) = \{R \mid \text{ for every active rule } R \text{ in } Q\}$. Follows the notion of multi-dimensional dynamic program updates for agents.

Definition 5. Let $m \in S$ be a state and $\Psi_\alpha^m = (\mathcal{T}, \mathcal{U})$ an agent α at state m. Let $\mathcal{T} = \langle D, \mathcal{P}_D, \mathcal{A}, \mathcal{R}_D \rangle$ and $\mathcal{U} = \{U_s \mid s \in S \text{ and } s > 0\}$. The **multi-dimensional dynamic program update for agent** α **at the state** m is $\bigoplus \Psi_\alpha^m = \langle P', \mathcal{A}, R' \rangle$ where: $P' = \bigcup_{v \in V} \gamma(v, 0, P_v) \cup \bigcup_{s \in S} (\Omega_1(Q_s)) \cup \mathit{IR} \cup \mathit{RR} \cup \mathit{UR} \cup \mathit{DR} \cup \mathit{CSR}(\alpha'_m) \cup \mathit{PRR} \cup \mathit{PDR} \cup \mathit{ER} \cup \mathit{EIR} \cup \mathit{RER}(\alpha'_m) \cup \mathit{RPR}$ and $R' = \bigcup_{v \in V} \gamma(v, 0, R_v) \cup \bigcup_{s \in S} (\Omega_2(Q_s)) \cup \mathit{PPR} \cup \mathit{PUR} \cup \mathit{PAR}(\alpha'_m)$, where $\beta(\alpha, U_s) = (\_, P_s)$, and $\sigma(s, P_s) = Q_s$, for every $s \in S$.

Note that P' is a normal logic program, that is, default atoms can only occur in bodies of clauses.

4 The Agent Cycle

In this section we briefly sketch the behaviour of an agent. We omit the details to keep the description general and abstract. Every agent can be thought of as a multi-dimensional abductive logic program equipped with a set of *inputs* represented as *updates*. The abducibles are (names of) *actions* to be executed as well as explanations of observations made. Updates can be used to solve the goals of the agent as well as to trigger new goals. The basic "engine" of the agent is the IFF proof procedure [10,6], executed via the cycle represented in Fig.3. Assume that $\beta(\alpha, U_s) = (L_1 \wedge \ldots \wedge L_n, P_s)$, $\sigma(s, P_s) = Q_s$ and $\Psi_\alpha^{s-1} = (\mathcal{T}, \{U_1, \ldots, U_{s-1}\})$.

The cycle of an agent α starts at state s by observing any inputs (projects from other agents) from the environment (step 1), and by recording them in the form of updates in the updating program U_s. Then, the proof procedure is applied for r units of time (steps 2 and 3) with respect to the abductive logic program $\bigoplus \Psi_\alpha^s$ obtained by updating the current one with the updating program U_s. The new goal is obtained by adding the goals $\mathit{not\,false} \wedge L_1 \wedge \ldots \wedge L_n$ and the active rules in $\Omega_2(Q_s)$ received from U_s to the current goal G. The amount of resources available in steps 2 and 3 is bounded by some amount r. By decreasing r the agent is more *reactive*, by increasing r the agent is more *rational*.

Afterwards (steps 4 and 5), the selectable projects are selected from G', the last formula of the derivation, and executed under the control of an abductive logic programming proof procedure such as ABDUAL in [3]. If the selected

```
Cycle($\alpha$,$r$,$s$,$\Psi_\alpha^{s-1}$,$G$)

1. Observe and record any input in the updating program $U_s$.
2. Resume the IFF procedure by propagating the inputs wrt. the
        program $\bigoplus \Psi_\alpha^s$ and the goal $G \wedge \mathit{not\,false} \wedge L_1 \wedge \ldots \wedge L_n \wedge \Omega_2(Q_s)$.
3. Continue applying the IFF procedure, using for steps 2 and 3 a
        total of $r$ units of time. Let $G'$ be the last formula in this derivation.
4. Select from $G'$ the projects that can be executed.
5. Execute the selected projects.
6. Cycle with ($\alpha$,$r$,$s+1$,$\Psi_\alpha^s$,$G'$).
```

Fig. 3. *The agent cycle*

project takes the form $j : C$ (meaning that agent α intends to update the theory of agent j with C at state s), then (once executed) the update $\alpha \div C$ will be available (at the next cycle) in the updating program of j. Selected projects can be thought of as outputs into the environment, and observations as inputs from the environment. From every agent's viewpoint, the environment contains all other agents. Every disjunct in the formulae derived from the goal G represents an *intention*, i.e., a (possibly partial) plan executed in stages. A sensible action selection strategy may select actions from the same disjunct (intention) at different iterations of the cycle. Failure of a selected plan is obtained via logical simplification, after having propagated `false` into the selected disjunct.

Integrity constraints provide a mechanism for constraining explanations and plans, and action rules for allowing a condition-action type of behaviour.

5 Conclusions and Future Work

Throughout this paper we have extended the *Multi-dimensional Dynamic Logic Programming* framework to include integrity constraints and active rules, as well as to allow the associated acyclic directed graph to be updatable. We have shown how to express, in a multi-agent system, each agent's viewpoint regarding its place in its perception of others. This is achieved by composing agents' view of one another in acyclic directed graphs, one for each agent, which can evolve over time by updating its edges. Agents themselves have their knowledge expressed by abductive generalized logic programs with integrity constraints and active rules. They are kept active through an observe-think-act cycle, and communicate and evolve by realizing their projects to do so, in the form of updates acting on other agents or themselves. Projects can be knowledge rules, active rules, integrity constraints, or abductive queries.

With respect to future work, we are following two lines of research. At the agent level, we are investigating how to combine logical theories of agents expressed over graph structures [13], and how to express preferences among the beliefs of agents [7]. At the multi-agent system level, we are investigating into the representation and evolution of dynamic societies of agents. A first step towards this goal is to allow the set of vertices of the acyclic directed graph to be also updatable, representing the birth (and death) of the society's agents. Our ongoing work and future projects is discussed in [2].

Acknowledgments We would like to thank José Alferes for all his comments. L. M. Pereira acknowledges the support of POCTI project FLUX. The work of J. A. Leite was also supported by PRAXIS Scholarship no. BD/13514/97.

References

1. J. J. Alferes, J. A. Leite, L. M. Pereira, H. Przymusinski and T. C. Przymusinski. *Dynamic Updates of Non-Monotonic Knowledge Bases.* Journal of Logic Programming 45(1-3), pages 43-70, 2000.
2. J. J. Alferes, P. Dell'Acqua, E. Lamma, J. A. Leite, L. M. Pereira, and F. Riguzzi. *A logic based approach to multi-agent systems.* ALP Newsletter, 14(3), August 2001.
3. J. J. Alferes, L. M. Pereira and T. Swift. *Well-founded Abduction via Tabled Dual Programs.* In Proc. of ICLP'99. MIT Press. 1999
4. M. Bozzano, G. Delzanno, M. Martelli, V. Mascardi and F. Zini, *Logic Programming and Multi-Agent System: A Synergic Combination for Applications and Semantics.* In *The Logic Programming Paradigm - A 25-Year Perspective*, pages 5-32, Springer 1999.
5. D. De Schreye, M. Hermenegildo and L. M. Pereira, *Paving the Roadmaps: Enabling and Integration Technologies.* Available from `http://www.compulog.org/net/Forum/ Supportdocs.html`
6. P. Dell'Acqua, L. M. Pereira, *Updating Agents.* In S. Rochefort, F. Sadri and F. Toni (eds.), Procs. of the ICLP'99 Workshop MASL'99, 1999.
7. P. Dell'Acqua, L. M. Pereira, *Preferring and Updating with Multi-Agents.* In 9th DDLP-WS. on "Deductive Databases and Knowledge Management", Tokyo, 2001
8. P. Dell'Acqua, F. Sadri, and F. Toni. *Combining introspection and communication with rationality and reactivity in agents.* In *Logics in Artificial Intelligence*, LNCS 1489, pages 17–32, Berlin, 1998. Springer-Verlag.
9. N. Jennings, K. Sycara and M. Wooldridge. *A Roadmap of Agent Research and Development.* In Autonomous Agents and Multi-Agent Systems, 1, Kluwer, 1998.
10. T. H. Fung and R. Kowalski. *The IFF proof procedure for abductive logic programming. J. Logic Programming*, 33(2):151–165, 1997.
11. R. Kowalski and F. Sadri. *Towards a unified agent architecture that combines rationality with reactivity.* In D. Pedreschi and C. Zaniolo (eds), Procs. of LID'96, pages 137-149, Springer-Verlag, LNAI 1154, 1996.
12. J. A. Leite, J. J. Alferes and L. M. Pereira, *Multi-dimensional Dynamic Logic Programming*, In Procs. of CLIMA'00, pages 17-26, July 2000.
13. J. A. Leite, J. J. Alferes and L. M. Pereira, *Multi-dimensional Dynamic Knowledge Representation.* In Procs. of LPNMR-01, pp 365-378, Springer, LNAI 2173, 2001.
14. J. A. Leite and L. M. Pereira. *Generalizing updates: from models to programs.* In Procs. of LPKR'97, pages 224-246, Springer, LNAI 1471, 1998.
15. S. Rochefort, F. Sadri and F. Toni, editors, *Proceedings of the International Workshop on Multi-Agent Systems in Logic Programming*, Las Cruces, New Mexico, USA,1999. Available from `http://www.cs.sfu.ca/conf/MAS99`.
16. I. Niemelä and P. Simons. *Smodels - an implementation of the stable model and well-founded semantics for normal logic programs.* In Procs. of the 4th LPNMR'97, Springer, July 1997.
17. F. Sadri and F. Toni. *Computational Logic and Multiagent Systems: a Roadmap*, 1999. Available from `http://www.compulog.org`.
18. The XSB Group. *The XSB logic programming system, version 2.0*, 1999. Available from `http://www.cs.sunysb.edu/~sbprolog`.

Enabling Agents to Update Their Knowledge and to Prefer

Pierangelo Dell'Acqua[1] and Luís Moniz Pereira[2]

[1] Department of Science and Technology
Campus Norrköping, Linköping University
Norrköping, Sweden
pier@itn.liu.se

[2] Centro de Inteligência Artificial - CENTRIA
Departamento de Informática
Faculdade de Ciências e Tecnologia
Universidade Nova de Lisboa
2829-516 Caparica, Portugal
lmp@di.fct.unl.pt

1 Introduction

In a previous paper [5] we presented a combination of the dynamic logic programming paradigm proposed by J. J. Alferes et al. [1,10] and a version of KS-agents proposed by Kowalski and Sadri [7]. In the resulting framework, rational, reactive agents can dynamically change their own knowledge bases as well as their own goals. In particular, at every iteration of an observe-think-act cycle, the agent can make observations, learn new facts and new rules from the environment, and then it can update its knowledge accordingly. The agent can also receive a piece of information that contrasts with its knowledge. To solve eventual cases of contradiction within the theory of an agent, techniques of contradiction removal and preferences among several sources can be adopted [8]. The actions of an agent are modeled by means of updates, inspired by the approach in [3]. A semantic characterization of updates is given in [1] as a generalization of the stable model semantics of normal logic programs [6]. Such a semantics is generalized to the three-valued case in [3], which enable us to update programs under the well-founded semantics.

J. J. Alferes and L. M. Pereira [2] propose a logic programming framework that combines two distinct forms of reasoning: preferring and updating. In particular, they define a language capable of considering sequences of logic programs that result from the consecutive updates of an initial program, where it is possible to define a priority relation among the rules of all successive programs. Updating and preferring complement each other in the sense that updates create new models, while preferences allow us to select among pre-existing models. Moreover, within the framework, the priority relation can itself be updated. In [2] the authors also define a declarative semantics for such a language.

In this paper we set forth an extension of the language introduced in [5] to allow agents to update their knowledge and to prefer among alternative choices.

In the resulting framework agents can, not only update their own knowledge, for example on advice from some other agent, but can also express preferences about their own rules. Preferences will allow an agent to select among alternative models of its knowledge base. In our framework, preferences can also be updated, possibly on advice from others. We exhibit examples to show how our approach functions, including how preferring can enhance choice of reactivity in agents. The declarative semantics of this approach to agents can be found in [4].

In the remaining of the paper, we will use the following example (taken from [2]) as a working example. This example, where rules as well as preferences change over time, shows the need to combine preferences and updates in agents, including the updating of preferences themselves.

Happy story. *(0) In the initial situation Maria is living and working everyday in the city. (1) Next, as Maria have received some money, Maria conjures up other, alternative but more costly, living scenarios, namely traveling, settling up on a mountain, or living by the beach. And, to go with these, also the attending preferences, but still in keeping with the work context, namely that the city is better for that purpose than any of the new scenarios, which are otherwise incomparable amongst themselves. (2) Consequently, Maria decides to quit working and go on vacation, supported by her increased wealth, and hence to define her vacation priorities. To wit, the mountain and the beach are each preferable to travel, which in turn gainsays the city. (3) Next, Maria realizes her preferences keep her all the while undecided between the mountain and the beach, and advised by a friend opts for the former.*

To keep the notation short, we follow some abbreviations: we let c stand for "living in the city", mt for "settling on a mountain", b for "living by the beach", t for "traveling", wk for "work", vac for "vacations", and mo for "possessing money".

2 Enabling Agents to Update Their Knowledge

This section introduces the language and concepts that allow agents to update their knowledge. Typically, an agent can hold positive and negative information. An agent, for example, looking at the sky, can observe that it is snowing. Then, after a while, when weather conditions change, the agent observes that it is not snowing anymore. Therefore, he has to update his own knowledge with respect to the new observed information. This implies that the language of an agent should be expressive enough to represent both positive and negative information. Knowledge updating refers not only to facts, as above, but also to rules which can also be overridden by newer rules which negate previous conclusions.

In order to represent negative information in logic programs, we need more general logic programs that allow default negation *not A* not only in premises of their clauses but also in their heads[1]. We call such programs generalized logic programs. It is convenient to syntactically represent generalized logic programs

[1] For further motivation and intuitive reading of logic programs with default negations in the heads see [1].

as propositional Horn theories. In particular, we represent default negation *not* A as a standard propositional variable. Propositional variables whose names do not begin with "*not*" and do not contain the symbols ":" and "$\div$" are called *objective atoms*. Propositional variables of the form *not* A are called *default atoms*. Objective atoms and default atoms are generically called *atoms*.

Agents interact by exchanging information: for instance, when an agent β communicates C to an agent α. Then, α may or may not believe C. In case α does believe it, α has to update its knowledge accordingly. This kind of agent interaction is formalized via the concepts of project and update. Propositional variables of the form $\alpha{:}C$ (where C is defined below) are called *projects*. $\alpha{:}C$ denotes the intention (of some agent β) of proposing the updating the theory of agent α with C. We assume that projects cannot be negated.

Propositional variables of the form $\beta \div C$ are called *updates*. $\beta \div C$ denotes an update that has been proposed by β of the current theory (of some agent α) with C. We assume that updates can be negated. A negated update *not* $\beta{\div}C$ in the theory of an agent α indicates that agent β did not have the intention of proposing the updating the theory of agent α with C.

Definition 1. *A generalized rule is a rule of the form* $L_0 \leftarrow L_1 \wedge \ldots \wedge L_n$ *(*$n \geq 0$*), where* L_0 *(*$L_0 \neq$ *false) is an atom and every* L_i *(*$1 \leq i \leq n$*) is an atom, an update or a negated update.*

Note that, according to the definition above, only objective atoms and default atoms can occur in the head of generalized rules.

Definition 2. *An integrity constraint is a rule of the form false* $\leftarrow L_1 \wedge \ldots \wedge L_n \wedge Z_1 \wedge \ldots \wedge Z_m$ *(*$n \geq 0, m \geq 0$*), where every* L_i *(*$1 \leq i \leq n$*) is an atom, an update or a negated update, and every* Z_j *(*$1 \leq j \leq m$*) is a project.*

Integrity constraints are rules that enforce some condition over the state, and therefore always take the form of denials (without loss of generality) in a 2-valued semantics. Note that generalized rules are distinct from integrity constraints and should not be reduced to them. In fact, in generalized rules it is of crucial importance which atom occurs in the head.

Definition 3. *A generalized logic program* P *is a set of generalized rules and integrity constraints.*

Definition 4. *A query* Q *takes the form* $?- L_1 \wedge \ldots \wedge L_n$ *(*$n \geq 1$*), where every* L_i *(*$1 \leq i \leq n$*) is an atom, an update or a negated update.*

Reactivity is a useful ability that agents must exhibit, e.g., if something happens, then the agent needs to perform some action. Agent's reactive behaviour is formalized via the notion of active rules. As active rules will be evaluated bottom-up, we employ a different notation for them to emphasize this aspect.

Definition 5. *An active rule is a rule of the form* $L_1 \wedge \ldots \wedge L_n \Rightarrow Z$ *(*$n \geq 0$*), where every* L_i *(*$1 \leq i \leq n$*) is an atom, an update or a negated update, and* Z *is a project.*

Active rules are rules that can modify the current state, to produce a new state, when triggered. If the body $L_1 \wedge \ldots \wedge L_n$ of the active rule is satisfied, then the project (fluent) Z can be selected and executed. The head of an active rule must be a project that is either internal or external. An *internal project* operates on the state of the agent itself, e.g., if an agent gets an observation, then it updates its knowledge, or if some conditions are met, then it executes some goal. *External projects* instead operate on the state of other agents, e.g., when an agent α proposes to update the theory of another agent β.

Example 1. Suppose that the underlying theory of Maria (represented with m) contains the following active rules:

$$\begin{array}{l} mo \Rightarrow m{:}not\ wk \\ b \Rightarrow m{:}goToBeach \\ t \Rightarrow p{:}bookTravel \end{array}$$

The heads of the first two active rules are project internal to Maria. The first states that if Maria has money, then she wants to update her own theory with *not wk*. The head of the last active rule is an external project: if Maria wants to travel, she proposes to book a travel to Pedro (represented with p).

We assume that for every project $\alpha{:}C$, C is either a generalized rule, an integrity constraint, an active rule or a query. Thus, a project can only take one of the following forms:

$$\begin{array}{l} \alpha{:}(L_0 \leftarrow L_1 \wedge \ldots \wedge L_n) \\ \alpha{:}(\mathit{false} \leftarrow L_1 \wedge \ldots \wedge L_n \wedge Z_1 \wedge \ldots \wedge Z_m) \\ \alpha{:}(L_1 \wedge \ldots \wedge L_n \Rightarrow Z) \\ \alpha{:}(?{-}L_1 \wedge \ldots \wedge L_n) \end{array}$$

Note that projects can only occur in the head of active rules and in the body of integrity constraints. Updates and negated updates can occur in the body of generalized rules and in the body of integrity constraints. For example, the integrity constraint $\mathit{false} \leftarrow A \wedge \beta{:}B$ in the theory of an agent α enforces the condition that α cannot perform a project $\beta{:}B$ when A holds. The active rule $A \wedge not \beta{\div}B \Rightarrow \beta{:}C$ in the theory of α states to perform project $\beta{:}C$ if A holds and α has not been proposed (by β) to update its theory with B.

When an agent α receives an update $\beta \div C$, α may or may not believe C depending on whether α trusts β. This behaviour, characterized at the semantic level in [4], can be understood with an axiom schema of the form:

$$C \leftarrow \beta \div C \wedge not\ \ \mathit{distrust}(\beta \div C)$$

in the theory of α.

3 Enabling Agents to Prefer

Preferring is an important ability that agents must exhibit to select among alternative choices. Consider for instance the following example.

Example 2. Let Q be the underlying theory of Maria.

$$Q = \left\{ \begin{array}{lr} c \leftarrow not\, mt \wedge not\, b \wedge not\, t & (1) \\ wk & (2) \\ vac \leftarrow not\, wk & (3) \\ mt \leftarrow not\, c \wedge not\, b \wedge not\, t \wedge mo & (4) \\ b \leftarrow not\, c \wedge not\, mt \wedge not\, t \wedge mo & (5) \\ t \leftarrow not\, c \wedge not\, mt \wedge not\, b \wedge mo & (6) \end{array} \right\}$$

As Q has a unique 2-valued model $\{c, wk\}$, Maria decides to live in the city (c). Things change if we add the rule mo (indicating that Maria possesses money) to Q. In this case, Q has four models: $\{c, mo, wk\}$, $\{mt, mo, wk\}$, $\{b, mo, wk\}$ and $\{t, mo, wk\}$, and thus c is no longer true. Maria is now unable to decide where to live.

To incorporate the ability of preferring, we extend the agent language to make it possible to express preferences among the rules of the agent itself. Let $<$ be a binary predicate symbol whose set of constants includes all the generalized rules.

Definition 6. *A priority rule is a generalized rule defining the predicate symbol* $<$.

It is assumed that the set of constants of $<$ does not include $<$ itself. $r_1 < r_2$ means that rule r_1 is preferred to rule r_2.

Definition 7. *A prioritized logic program P is a generalized logic program possibly containing priority rules.*

The definition of prioritized logic program generalizes the one given in [5] to allow for priority rules. The intuition is that the program P formalizes the theory of an agent, and the priority rules express preferences among rules in P.

Example 3. Let P be the following prioritized logic program:

$$P = Q \cup \{mo\} \cup \left\{ \begin{array}{l} 1 < 4 \leftarrow wk \\ 1 < 5 \leftarrow wk \\ 1 < 6 \leftarrow wk \\ 4 < 6 \leftarrow vac \\ 5 < 6 \leftarrow vac \\ 6 < 1 \leftarrow vac \end{array} \right\}$$

For simplicity, we adopt unique numbers for rules, instead of the rules themselves, in the priority relation $<$. The first three priority rules in P say that rule 1 is preferable to the rules 4, 5 and 6 if wk holds. If vac holds, then the rules 4 and 5 are both preferable to rule 6 which in turn is preferable to rule 1. As wk holds while vac does not, the first three priority rules are used to characterise the preferred model: $\{c, mo, wk\}$. Thus, by reducing the number of models of a program, priority rules allow us to select among several alternative models. Not always the priority rules allow us to select exactly one model. This may happen when the priority rules do not specify a complete linear order among the alternative choices. Suppose for instance that wk is false, then vac would hold and P would have two models: $\{mt, mo, wk\}$ and $\{b, mo, wk\}$.

As priority rules are generalized rules, preferences can also be updated. For example, we can update the priority rules of the program P in the previous example by updating with the rule $4{<}5{\leftarrow}vac$. In this case, P will have a unique model (also in case *vac* holds).

4 Agents and Multi-agent Systems

This section presents the conception of agent and of multi-agent system. The initial knowledge of an agent is modelled by the notion of initial theory.

Definition 8. *The initial theory $\mathcal{T}$ of an agent α is a pair (P, R), where P is a prioritized logic program and R is a set of active rules.*

P formalizes the initial knowledge state of the agent, and R characterizes its reactive behaviour. The knowledge of an agent can dynamically evolve when the agent receives new knowledge, albeit by self-updating rules. The new knowledge is represented in the form of an updating program.

Definition 9. *Let $\mathcal{M}$ be a multi-agent system (defined below). An updating program U is a finite set of updates such that if an update $\alpha{\div}C \in U$ then Ψ_α is an agent of $\mathcal{M}$.*

An updating program contains the updates that will be performed on the current knowledge state of the agent. To characterize the evolution of the knowledge of an agent we need to introduce the notion of sequence of updating programs. Let $S = \{0, 1, \ldots, m\}$ be a set of natural numbers. We call the elements $s \in S$ *states*. A *sequence of updating programs* $\mathcal{U} = \{U^s \mid s \in S \text{ and } s > 0\}$ is a set of updating programs U^s superscripted by the states $s \in S$.

Definition 10. *Let $s \in S$ be a state. An agent α at state s, written as Ψ_α^s, is a pair $(\mathcal{T},\mathcal{U})$, where $\mathcal{T}$ is the initial theory of α, and $\mathcal{U} = \{U^1, \ldots, U^s\}$ is a sequence of updating programs. If $s = 0$, then $\mathcal{U} = \{\}$.*

An agent α at state 0 is defined by its initial theory and an empty sequence of updating programs, that is $\Psi_\alpha^0 = (\mathcal{T}, \{\})$. At state 1, α is defined by $(\mathcal{T}, \{U^1\})$, where U^1 is the updating program containing all the updates that α has received at state 1, either from other agents or as self-updates. In general, an agent α at state s is defined by $\Psi_\alpha^s = (\mathcal{T}, \{U^1, \ldots, U^s\})$, where each U^i is the updating program containing the updates that α has received at state i.

Definition 11. *A multi-agent system $\mathcal{M} = \{\Psi_{\alpha_1}^s, \ldots, \Psi_{\alpha_n}^s\}$ at state s is a set of agents $\alpha_1, \ldots, \alpha_n$ each of which at state s.*

Note that the definition of multi-agent system characterizes a static society of agents in the sense that it is not possible to add/remove agents from the system, and all the agents are at one common state.

To begin with, the system starts at state 0 where each agent is defined by $\Psi_\alpha^0 = (\mathcal{T}_\alpha, \{\})$. Suppose that at state 0 an agent β proposes an update of the knowledge state of agent α with C by triggering the project $\alpha : C$. Then, at the next common state α will receive the update $\beta \div C$ indicating that an update has been proposed by β. Thus, at state 1, α will be $(\mathcal{T}_\alpha, \{U^1\})$, where $U^1 = \{\beta \div C\}$ if no other updates occur for α.

Example 4. This example formalizes the happy story as described in Section 1. Let the initial theory $\mathcal{T} = (P, R)$ of Maria be:

$$R = \left\{ \begin{array}{l} mo \Rightarrow m{:}not\ wk \\ b \Rightarrow m{:}goToBeach \\ mt \Rightarrow m{:}goToMountains \end{array} \right\}$$

$$P = \left\{ \begin{array}{ll} c \leftarrow not\, mt \wedge not\, b \wedge not\, t & (1) \\ wk & (2) \\ vac \leftarrow not\ wk & (3) \\ mt \leftarrow not\ c \wedge not\, b \wedge not\, t \wedge mo & (4) \\ b \leftarrow not\ c \wedge not\ mt \wedge not\, t \wedge mo & (5) \\ t \leftarrow not\ c \wedge not\ mt \wedge not\, b \wedge mo & (6) \\ 1 < 4 \leftarrow wk & \\ 1 < 5 \leftarrow wk & \\ 1 < 6 \leftarrow wk & \\ 4 < 6 \leftarrow vac & \\ 5 < 6 \leftarrow vac & \\ 6 < 1 \leftarrow vac & \end{array} \right\}$$

At state 0, suppose that Maria receives the information from some agent l that she has received some money, that is, $U^1_m = \{l{\div}mo\}$. Thus, as Maria does not distrust agent l, at state 1 she believes mo and therefore she proposes the internal update $m{:}not\ wk$ by triggering the first active rule in R. At state 2 Maria does not work and therefore decides to take a vacation. Unfortunately, the theory of Maria has two models:

$$\{mt, vac, mo, 4 < 6, 4 < 1, 5 < 6, 5 < 1, 6 < 1\}$$
$$\{b, vac, mo, 4 < 6, 4 < 1, 5 < 6, 5 < 1, 6 < 1\}$$

and therefore Maria is unable to take any action neither living by the beach nor settling up on a mountain. Suppose now that Maria is advised by a friend f to prefer mountains to beaches, that is, $U^2_m = \{f{\div}(4{<}5{\leftarrow}vac)\}$. At state 3 the theory of Maria has a single preferred model:

$$\{mt, vac, mo, 4 < 5, 4 < 6, 4 < 1, 5 < 6, 5 < 1, 6 < 1\}$$

and Maria can happily settle up on the mountains.

5 Conclusion and Future Work

We have presented a logical framework of a multi-agent system in which each agent can communicate with and update other agents, can react, and is able to prefer. In the near future, we plan to develop a proof procedure for updating and preferring reasoning and to prove it correct and complete with respect to the declarative semantics presented in [4]. We are also investigating possible generalizations to the multi-agent system to make it not synchronous, and dynamic in a way that the agents can enter and leave the system.

References

1. J. J. Alferes, J. A. Leite, L. M. Pereira, H. Przymusinski, and T. C. Przymusinski. Dynamic updates of non-monotonic knowledge bases. *J. Logic Programming*, 45(1-3):43–70, 2000.
2. J. J. Alferes and L. M. Pereira. Updates plus preferences. In M. O. Aciego, I. P. de Guzmn, G. Brewka, and L. M. Pereira, editors, *Logics in AI, Procs. JELIA'00*, LNAI 1919, pages 345–360, Berlin, 2000. Springer.
3. J. J. Alferes, L. M. Pereira, H. Przymusinska, T. C. Przymusinski, and P. Quaresma. Preliminary exploration on actions as updates. Proc. of Joint Conf. on Declarative Programming (APPIA-GULP-PRODE'99).
4. P. Dell'Acqua and L. M. Pereira. Preferring and updating with multi-agents. Deductive Databases and Knowledge Management (DDLP'2001). To appear.
5. P. Dell'Acqua and L. M. Pereira. Updating agents. In S. Rochefort, F. Sadri and F. Toni (eds.), Procs. of the ICLP'99 Workshop on Multi-Agent Systems in Logic (MASL'99), 1999.
6. M. Gelfond and V. Lifschitz. The stable model semantics for logic programming. In R. Kowalski and K. A. Bowen, editors, *ICLP'88*, pages 1070–1080. MIT Press, 1988.
7. R. A. Kowalski and F. Sadri. Towards a unified agent architecture that combines rationality with reactivity. In D. Pedreschi and C. Zaniolo, editors, *Logic in Databases, Intl. Workshop LID'96*, LNCS 1154, pages 137–149. Springer, 1996.
8. E. Lamma, F. Riguzzi, and L. M. Pereira. Strategies in combined learning via logic programs. *Machine Learning*, 38(1/2):63–87, 2000.
9. J. A. Leite, J. J. Alferes, and L. M. Pereira. Minerva - A Dynamic Logic Programming Agent Architecture. To appear in: ATAL01 - 8th Int. Workshop on Agent Theories, Architectures, and Languages, 2001.
10. J. A. Leite and L. M. Pereira. Iterated logic program updates. In J. Jaffar, editor, *Proc. Joint Intl. Conf. Symp. on Logic Programming 1998*, pages 265–278, Cambridge, Mass., 1998. MIT Press.
11. A. Y. Levy and D. S. Weld. Intelligent internet systems. *Artificial Intelligence*, 118:1–14, 2000.

Modelling Agent Societies: Co-ordination Frameworks and Institutions

Virginia Dignum [1,2], Frank Dignum[2]

[1] Achmea
PO Box 866, 3700 AW Zeist, The Netherlands
virginia.dignum@achmea.nl
[2] University Utrecht
Institute for Information and Computing Sciences
PO Box 80.089, 3508TB Utrecht, The Netherlands
dignum@cs.uu.nl

Abstract. Organisations can be defined as a set of entities regulated by mechanisms of social order and created by more or less autonomous actors to achieve common goals. Multi-agent systems are a natural choice to design organisational systems due to the proactive and autonomous behaviour of agents. However, in business environments it is necessary to consider the behaviour of the global system and the collective aspects of the domain. In this paper, we argue that multi-agent systems should be designed around organisational co-ordination frameworks that reflect the co-ordination structures of the particular organisation. As in human societies, we argue that norms and institutions are a way for agent societies to cope with the challenge of social order. Through institutions, conventions and interaction patterns for the co-ordination of agents can be specified, monitored and managed.

Keywords: Agent societies, co-ordination, institutions, virtual organisations

1. Introduction

In an increasing number of domains, organisations need to work together in transactions, tasks or missions. Work relationships between people and enterprises are also shifting, from the 'job-for-life' paradigm to project-based virtual enterprises in which people and organisations become independent contractors. Furthermore, there is often a decentralised ownership of data, expertise, control and resources involved in business processes. Often, multiple, physically distributed organisations (or parts hereof) are involved in one business process. Each organisation, or part of an organisation, attempts to maximise its own profit within the overall activity. Different groups within organisations are relatively autonomous, in the sense that they control how their resources are created, managed or consumed, and by whom, at what cost, and in what time frame. There is a high degree of natural concurrency (many interrelated tasks and actors are working simultaneously at any given point of the business process) which makes it imperative to be able to monitor and manage the overall business process (e.g. total time, total budget, etc.). The above considerations show an increasing need for transparency in the representation and implementation of business processes. However, the fact that business processes are highly dynamic and unpredictable makes it difficult to give a complete a priori specification of all the activities that need to be performed, which are their knowledge needs, and how they should be ordered.

An organisation can be seen as a set of entities regulated by mechanisms of social order and created by more or less autonomous actors to achieve common goals. Because of the proactive and autonomous behaviour of agents it is natural to design organisation systems using agent societies that mimic the behaviour and structure of human organisations [22]. Agent societies represent the interactions between agents and are as such the virtual counterpart of real-life societies and organisations. Agents model specific roles in the society and interact with others as a means to accomplish their goals. This perspective makes the design of the system less complex since it reduces the conceptual distance between the system and the real-world application it has to model. Therefore, agent societies are an effective platform for virtual organisations because they provide mechanisms to allow organisations to advertise their capabilities, negotiate their terms, exchange rich information, and synchronise processes and workflow at a high-level of abstraction [18].

Business environments must consider the behaviour of the global system and be able to incorporate collective characteristics of an organisation such as stability over time, some level of predictability, and clear commitment to aims and strategies. However, typically, agents are assumed to pursue their own individual goals and global behaviour emerges from individual interactions. Existing architectures, behavioural strategies and models for group formation often assume this individualist perspective, which is not suitable for the representation of collective characteristics of an organisation.

In this paper, we argue that multi-agent systems developed to model and support organisations must be based on co-ordination frameworks that mimic the structure of the particular organisation. Methodologies for designing such multi-agent systems have to be able to describe and apply different types of co-ordination models. As in human societies, we argue that norms and institutions are a way for agent societies to cope with the challenge of social order. Agents act autonomously according to their own goals and capabilities. Institutions are needed to enforce the global behaviour of the society and assure that the global goals of the society are met. Different co-ordination models have different needs in terms of how institutions can manage them and consequently which type of roles are present in the institution and which should be the capabilities of the agents fulfilling those roles.

The paper is organised as follows. In section 2 we introduce a model for agent societies that is based on the structural characteristics of an organisation and supported by different co-ordination frameworks. The role of institutions in the engineering of agent societies is described in section 3. In section 4, the characteristics of the different frameworks are described in more detail. Practical applications of this model being developed at Achmea are described in section 5. Finally, in section 6 we present some conclusions and indicate directions for future work.

2. Organisational Multi-agent Systems

There is a rising awareness that multi-agent systems and cyber-societies can best be understood and developed if they are inspired by human social phenomena [1, 5, 23]. Organisations can be seen as sets of entities regulated by mechanisms of social order and created by more or less autonomous actors to achieve common goals. Multi-agent systems that model and support organisations should therefore be based on co-ordination frameworks that mimic the structure of the particular organisation and be able to dynamically adapt to changes in organisation structure, aims and interactions. The structure of the organisation determines important autonomous activities that must be explicitly organised into autonomous entities and relationships in the conceptual model of the agent society [11].

In a business environment, the behaviour of the global system and the collective aspects of the domain, such as stability over time, predictability and commitment to aims and strategies, must be considered. Organisations are expected to form a coherent, stable system that realises the objectives for which it was designed. When multi-agent systems, or **agent societies**, are considered from an organisational point of view, the concept of desirable social behaviour becomes of utmost importance. That is, from the organisational point of view, the behaviour of individual agents in a society should be understood and described in relation to the social structure and overall objectives of the society. However, until recently, multi agent systems are mainly viewed from an individualistic perspective, that is, as aggregations of agents that interact with each other [13]. In this view looks at the behaviour of multi-agent systems from the perspective of the agent itself, in terms of how an agent can affect the environment or be affected by it.

Open societies assume that participating agents are designed and developed outside the scope and design of the society itself and therefore the society cannot rely on the embedding of organisational and normative elements in the intentions, desires and beliefs of participating agents but must represent these elements explicitly.

The above considerations lead to the following requirements for engineering methodologies for agent societies:

- Agent societies must include formalisms for the description, construction and control of the organisational and normative elements of a society (roles, norms and goals) instead of just agent states [1, 23].
- The methodology must provide mechanisms to describe the environment of the society and the interactions between agents and the society, and to formalise the expected outcome of roles in order to verify the overall animation of the society.
- The organisational and normative elements of a society must be explicitly specified since an open society cannot rely on its embedding in the intentions, desires and beliefs of each agent [7, 17]
- Methods and tools are needed to verify whether the design of an agent society satisfies its design requirements and objectives [15].
- The methodology should provide building directives concerning the communication capability and ability to conform to the expected role behaviour of agents participating in the society.

One last point is that in order to facilitate the development of organisation oriented multi-agent systems it is important to relate to the organisational perception of the domain. That is, a common ground of understanding must be found between agent engineers and organisational practitioners. In our opinion co-ordination is an ideal candidate. In one hand, organisational science and economics have since long researched co-ordination and organisational structures. Relationships between and within organisations are developed for the exchange of goods, resources, information and so on. Depending on transaction costs and interdependent relations, different co-ordination models (market, hierarchy or network) are possible. On the other hand, co-ordination is one of the cornerstones of agent societies and is considered an important problem inherent to the design and implementation of MAS [2]. However, the implications of the co-ordination model for the agent society architecture and design method have usually not been considered. So far, research about co-ordination in MAS has been mainly limited to the study of technical aspects of co-ordination, such as control and planning. In many cases the social organisation is left implicit in the design of the agent society. An agent society model that incorporates co-ordination issues related to the organisational perspective of the domain will thus facilitate the introduction of multi-agent systems in organisations. Co-ordination forms therefore the basis for the model for agent societies introduced in this paper. The following notions are core concepts in our model:

- **Agents** are the inhabitants of the agent society that interact with each other using the communication framework. Agents are designed outside the scope of the society, and may have their own goals and behaviour rules. Every agent within the society must adopt some role(s).
- **Roles** are patterns of behaviour. Roles are described in the society model in terms of externally perceived behaviour
- **Rules** or **constraints** describe the desired behaviour of agents in the society and its consequences in terms of sanctions, rewards and limitations.
- **Communication framework** describes the interaction between agents. It includes the description of the society ontology (vocabulary understood within the society), the communication language (intentions and utterances) and the representation language for domain content.
- **Goals** are the overall objectives of the society

As described before, the design of organisation-oriented multi-agent systems must account for the representation and management of normative aspects of the society and incorporate collective characteristics of an organisation such as stability over time, some level of predictability, and clear commitment to aims and strategies. Human societies have successfully coped with similar issues through the use of institutions that monitor behaviour and enforce social laws. Therefore our agent society model consists of two layers. The institutional layer, or **institution**, provides the social and institutional backbone of the society and are the place where social norms and rules are explicitly specified. Institutional agent roles are designed to enforce the social behaviour of agents in the society and assure the achievement of global goals of the society. The **operational layer** models the overall objectives and intended action of the society and is therefore domain dependent. Interaction between agents in the operational level is not necessarily bound by the institution, and agents are free to act according to their own objectives. However, in order to join the society agents must commit themselves to the social rules described and enforced by the institution.

3. The Role of Institutions

Usually human organisations and societies use norms and conventions to cope with the challenge of social order. Norms and conventions specify the behaviour that society members are expected to conform to and are suitable for decentralised control. In most societies, norms are backed by a variety of social institutions that enforce law and order (e.g. courts, police), monitor for and respond to emergencies (e.g. ambulance system), prevent and recover from unanticipated disasters (e.g. coast guard, fire-fighters), etc. In this way civilised societies allow citizens to utilise relatively simple and efficient rules of behaviour, offloading the prevention and recovery of many problem types to social institutions that can handle them efficiently and effectively by virtue of their economies of scale and widely accepted legitimacy. Successful human institutions achieve sustainability of citizens and increase the welfare of the society as a whole. Several researchers have recognised that the design of agent societies can benefit from abstractions analogous to those employed by our robust and relatively successful societies and organisations. There is a growing body of work that touches upon the concepts of norms and institutions in the context of multi-agent systems (cf. [9, 10, 12]).

The benefit of an institution resides in its potential to lend legitimacy and security to its members by establishing norms. The electronic counterpart of the physical institution does a similar task for software agents: it can engender trust through certification of an agent and by the guarantees that it provides to back collaboration. However, the electronic institution can also function as the independent place in which al types of agent independent information about the interaction between the agents within the society is stored. E.g. it defines the message types that can be used by the agents in their interactions, the rules of encounter, etc. In general, institutions enable to:

- Specify the co-ordination structure that is used
- Describe exchange mechanisms of the agent society
- Determine interaction and communication forms within the agent society
- Facilitate the perception of individual agents of the aims and norms of an agent society
- Enforce the organisational aims of the agent society

In our approach we consider that an agent society consists of two layers: one is facilitation-oriented and the other goal-oriented. The institution acts as mediator and animator for the members, who bring various skills and services, and customers (or groups of customers) who bring their problems and requirements. The most important service the institution provides is to regulate the interaction between members. Because the way interaction between agents happens depends on the co-ordination model, institutions will need to be defined differently for each co-ordination model.

We have shown above that co-ordination models provide a setting for agent societies by setting out the goals of the society and the roles (what you can do) need to achieve those goals. Institutions will enforce this model by setting out the scenes (where you can do it) and protocols (what you can say) for interaction in the society. This defines how agents can interact with the institution or with other agents in the society. The whole point of institutions is for the additional services it can provide and the trust and guarantees that are established through the institution's credibility and norms.

Looking at the structure of organisations we can anticipate the types of interaction involved in interacting in a particular co-ordination model. Thus, an institution defines a performative structure and a dialogical framework, by which we mean, it prescribes the actions members can take and when and where to perform those actions, and determines the form of conversations between members. Therefore, the way norms and conventions are specified and enforced in a society depends on the co-ordination model. In hierarchies, norms and conventions can be embedded in the power relations. These relations determine which agent can demand an action from which other agent or which agent has priority over the resources. The controlling agent is supposed to uphold the norms of the society by managing the sub-ordinate agents according to them. In markets, norms and conventions are for a large part embedded in the market mechanism chosen. E.g. the auction mechanisms try to ensure that all agents get an opportunity to require a resource relative to their private value for that resource. Cheating by over- or underbidding does not lead to any benefit for the agent and thus is prevented by the mechanism itself. In network models explicit roles are defined to 'represent' the institutions that enforce monitoring and trust, and trace the fulfilment of contracts. Some examples of these roles will be given in the next section.

4. Co-ordination Models

We identify three basic co-ordination types of agent societies following on the classification of organisations used in organisational theory. Hence, co-ordination of agent societies follows a market, network or hierarchy model. Each co-ordination model determines a different framework for agent societies that describe the institutional layer of the society. The institutional layer must describe institutional roles, the way interactions between roles are organised and the way the interface between the society and the 'outside world' is defined. That is, the co-ordination model determines the institutional roles, social norms and interaction forms in the society.

In **markets**, agents are self-interested (determine and follow their own goals) and value their freedom of association and own judgement above security and trust issues. **Network** organisations are built around general patterns of interaction or contracts. Relationships are dependent on clear communication patterns and social norms. Agents in a network society are still self interested but are willing to trade some of their freedom to obtain secure relations and trust. Finally, in a **hierarchy** interaction lines are well defined and the facilitation level assumes the function of global control of the society and co-ordination of interaction with the outside world. Table 1 gives an overview of the characteristics of different agent societies.

The characteristics and requisites for each role determine the required capabilities of agents fulfilling the role in terms of its communicative and reasoning capabilities. For example, agents acting in a network are expected to negotiate their interaction procedures and are motivated by mutual interest. This means such agents will be required to be able to reason about other agents and need to possess 'heavy' negotiation algorithms. On the other hand, members of a hierarchical society follow pre-determined communication lines and have limited need for negotiation, thus agents fulfilling hierarchical roles can be much simpler in terms of communication and negotiation capabilities.

Table 1. Characteristics of agent societies

	Market	**Network**	**Hierarchy**
Type of society	Open	Trust	Closed
Members 'values'	Self interest	Mutual interest	Dependency
Society purpose	Exchange	Collaboration	Production
Interaction	Interaction is based on standards; communication concerns exchange only	Both interaction and exchange procedures can be negotiated	Specified on design

In order to be able to assign roles to agents, the society model must be able to make some assumptions on the capabilities of the agent. However, since open societies are based on the principle that participating agents are developed independently from the society, it is not possible to make too many assumptions on the specific architecture of agents. We use a generic agent model as a basis for our assumption on agents. This model is based on the work of [4]. This model makes no demands on the way internal agent components are designed, but assumes that agents will in some way be able to use the indicated capabilities. Agent engineers are free to design their agents' internal components in different ways, and even do without some of the components. The description of roles in the society model refers to this agent model and describes the society expectations on the capabilities of agents that perform the role.

We have developed a methodology (described in more detail in [11]) for the design of agent societies based on co-ordination structures. The aim of the methodology is to provide generic facilitation and interaction frameworks for agent societies that implement the functionality derived from the co-ordination model applicable to the problem domain. We can compare this process to the design a generic enterprise model including roles as accountants, secretaries and managers, as well as their job descriptions and relationships, and then extending it with the functions necessary to achieve the objectives of the given enterprise. These are, for example, designers and carpenters if the firm is going to manufacture chairs, and programmers and system analysts if the enterprise is a software house.

4.1. Roles in the Market Co-ordination Model

The main goal of a market is to facilitate exchange between agents. In a market model, agents are self-interested (determine and follow their own goals), represent (or provide) services and/or competencies and compete to perform tasks leading to the satisfaction of their own individual objectives. Agents are usually assumed to be heterogeneous and the negotiation rules are fixed (for example Contact Net or Dutch auction). Interaction in markets occurs through communication and negotiation with the market rules.

Co-ordination through a market mechanism is particularly well suitable for situations in which resources can be described easily or are commoditised, there are several agents offering the same (type) of resources and several agents that need them. Besides obvious e-commerce applications, the market architecture is also a good choice to model product or service allocation problems. Being self-interested, agents will first try to solve their own local problem, and then agents can potentially negotiate with other agents to exchange services or goods in shortage or in excess. Agent societies based on the market model have been used to represent virtual enterprises [19]. Facilitation roles necessary for the organisation of a market model are:

- **Identification**: has the task of registering members of the society. Can also receive requests from matchmakers or bankers
- **Matchmaker**: keeps track of agents in the system, their needs and possibilities and mediates in the matching of demand and supply of goods or services. Depending on the domain, the task of a matchmaker can be a simple unification algorithm or a complex fuzzy matching algorithm. Matchmakers must be able to receive requests from agents and contact possible partners. Depending on the domain, this capabilities can be just a simple message *request(buyer?, product, price)* or *announce(seller, product, price)* or it can involve more general communication determining the requirements on both products and potential partner. Furthermore, matchmakers need to have knowledge of current sellers and requests in the society. I.e. they need to maintain a kind of yellow guide.
- **Banking**: define ways to value the goods to be exchanged and determine profit and fairness of exchanges. A banking service builds confidence for customers as well as offers guarantees to the members of the society. Bankers must be able to receive requests from agents wishing to register themselves (open an account) or wishing to get information on other agents, and need to keep knowledge on their clients

4.2. Roles in the Hierarchy Co-ordination Model

Hierarchies co-ordinate the flow of resources or information by controlling and directing it at a central point in the managerial hierarchy. Interaction and design are determined by managerial decisions and achievement of global goals is most critical. Demand parties do not select a supplier from a group of potential suppliers: they simply work with a predetermined one. In hierarchical systems, each agent controls a statically defined sub-hierarchy (possibly empty), in many cases an administrative domain of some kind. Environments where the workflow is fixed and cases are repetitive, such as in automated manufacturing are well suited to the hierarchical model. In such systems, reliable control of resources and information flow requires central entities that manage local resources and data but also need quick access to global ones. Hierarchical models of agents have been used to model information agents ([6]) and the management of communication networks ([14]).

In a hierarchical co-ordination model, agents at facilitation level are mainly dedicated to the overall control and optimisation of the system activities. Sometimes, these facilitation activities are concentrated in one agent, typically the 'root' agent of the hierarchy. Facilitation roles necessary to the organisation of a hierarchy are:

- **Controllers**: monitor and orient the overall performance of the system or of a part of the system. Autonomous agents have local perspective and their actions are determined by its local state. Therefore, in a hierarchical co-ordination model it is necessary to have an agent whose role is to control the overall performance of the system.

- **Interface agents:** are responsible for the communication between the system and the ‘outside world’. In this architecture communication lines between agents are predefined. Furthermore, agents are usually not free to enter or leave the system. Therefore communication with the outside world must be regulated at the facilitation level.

4.3. Roles in the Network Co-ordination Model

Networks are coalitions of self-interested agents that agree to collaborate to achieve a mutual goal. Agents in a network society are self-interested but are willing to trade some of their freedom to obtain secure relations and trust. Instead of a direct exchange as in markets, agents in a network model are willing to trade their services in exchange for later or soft rewards (such as a increase of prestige). Network co-ordination models are built around general patterns of interaction or contracts. Relationships are dependent on clear communication patterns and social norms. Co-ordination is achieved by mutual interest, possibly using trusted third parties, and according to well-defined rules and sanctions. These coalitions have been studied in the area of game theory and Distributed Artificial Intelligence (DAI) [20]. Dellarocas introduces the concept of Contractual Agent Societies (CAS) as a model for developing agent societies [7]. Network co-ordination models provide an explicit shared context, describing rules and social norms for interaction and collaboration. The society is responsible to make its rules and norms known to potential members. Agents in a network society enter a social contract with the society in which they commit themselves to act within and according to the norms and rules of the society.

At the facilitation level of a network, agents monitor, register and help others form contracts, introduce (teach) new agents to the rules of the market and keep track of the reputation of agents. Furthermore, they keep and enforce the ‘norms’ of the agent community and ensure interaction. Roles at facilitation level in networks are:

- **Matchmaker**: keeps track of agents in the system, their needs and possibilities and mediates in the matching of demand and supply of goods or services. In the network co-ordination domain, the matching of supply and demand is usually more complex than in markets, because long-term interests have to be taken into account. Therefore, matchmakers will need to use, for instance, fuzzy matching algorithms, or multi-attribute matching to be able to perform their tasks. As in markets, matchmakers must be able to receive requests from agents and contact possible partners and need to keep knowledge of current offers and requests in the society.
- **Gatekeeper:** is responsible for accepting and introducing new agents to the market. Agents entering the marketplace must be informed about the possibilities and capabilities of the market. Gatekeepers negotiate the terms of a social contract between the applicant and the members of the market.
- **Notary:** register collaboration contracts between agents.
- **Monitoring agents:** are trusted third parties that keep track of the execution of collaboration contracts between agents.

5. Applications

The framework described in this paper can be applied to very distinct problem domains, because it concentrates on the organisational elements of the agent societies. At Achmea, a financial and insurance holding organisation operating mainly in the Netherlands, the ideas described in this paper are being applied to the development of a system for support of knowledge sharing (K-Exchange). This project is further described below. Other plans for application this framework include the development of a mediation system in the area of secondary healthcare co-ordination (CareMarket). Although both projects are still in a initial phase and no results are as yet available, the models developed illustrate the possibilities of the different co-ordination frameworks and the use of institutions

CareMarket

The aim of CareMarket, a community care project is to provide Achmea clients with extra (unskilled) care services, which are not covered by professional organisations, or for which there are long waiting lists. The project is inspired by the LETS concept and based on non-monetary trading concepts. Matching of supply and demand in this kind of situations is not trivial. The fulfilment of a demand usually requires the co-ordination of several suppliers, suppliers are voluntaries and usually of a very limited and constrained range of services. Furthermore, it is desirable to keep a continuity of relationships between suppliers and clients (people tend to develop friendship relations with their care tenders / care takers and do not really appreciate to see a new face every day). This pilot is in a very initial phase of development but there is already a clear realisation that the institutional framework described in this paper will be directly applicable to the development of an agent-based simulation prototype. The evaluation of the system through the simulated institution populated with intelligent agents, representing suppliers and clients, will provide insights and support to the eventual deployment of a real community pilot.

Knowledge Exchange Network

The objective of the Knowledge Exchange Network project is to support non-life insurance experts to exchange knowledge with each other, in a way that preserves the knowledge, rewards the knowledge owner and reaches the knowledge seeker in a just-in-time, just-enough basis. Current users of the pilot project are project managers, product developers, actuaries in the Non-life group of Achmea but in the future it will be extended to other people (e.g. call centre employees) and groups. Members of the network have lots of knowledge, which is greatly valuable and useful to each other. So, one of the main tasks of the Knowledge Exchange Network is to support and encourage their contacts. Experience shows that any technological support for knowledge exchange greatly improves if users feel they know and can trust each other. Therefore, the Knowledge Management activities at the Non-life group consist of two parts: face-to-face workshops with the aim of getting people to know each other, share their experiences and extend their knowledge and a virtual network, aiming both at a knowledge repository and at the support of communication and collaboration.

For the share support module, an agent society is being developed using the framework based design method described in this paper. In this society, both knowledge seekers as knowledge owners want to be able to decide on trade partners and conditions. Sharing is not centrally controlled but greatly encouraged by the management. The best-suited partner, according to each participant's own conditions and judgement, will get the 'job'. However, factors such as privacy, secrecy and competitiveness between brands and departments may influence the channels and possibilities of sharing and must thus be considered.

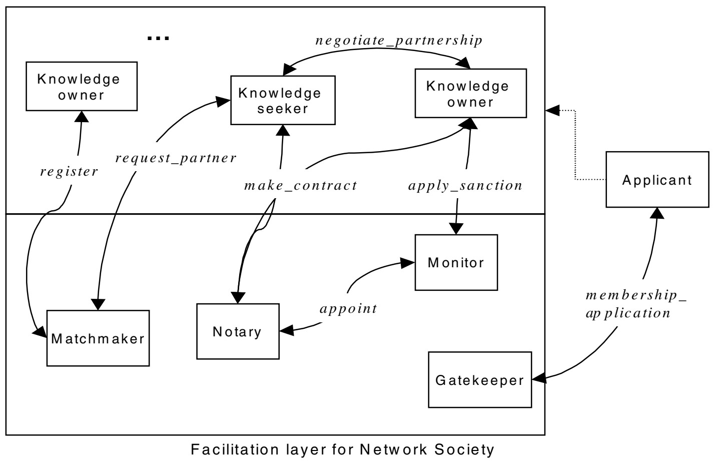
The requirements for the system identify a distributed system where different actors, acting autonomously on behalf of a user, and each pursuing its own goals, need to interact in order to achieve their goals. Communication and negotiation are paramount. Furthermore, the number and behaviour of participants cannot be fixed a priori and the system can be expected to expand and change during operation, both in number of participants as in amount and kind of knowledge shared. These characteristics indicate a situation for which the agent paradigm is well suited and therefore the methodology we propose can be applied.

Considering the requirements, the network model is the most appropriate for this situation. The aim is to design an exchange society restricted to selected participants with the global goal of supporting collaboration and synergy, and in this way meet the organisation requirements. Participants are aware of and collaborative with this requirement but also have their own objectives and constraints. Participants wish to be free to determine their own exchange rules and to be assured that there is control over who are the other participants in the environment.

Due to space limitations, we cannot describe the complete system in this paper. In the following we will describe some of the roles and interactions. Having decided for a network structure, the roles of matchmaker, notary, monitor, and gatekeeper follow naturally from the application of the framework. From the domain requirements the roles of knowledge owner and knowledge seeker can be deduced. The 'goods' to be exchanged are the contents of the knowledge repository, that is, (XML) documents representing knowledge about reports, people, applications, web sites, projects, questions, etc.[1] Figure 1 shows a fragment of the architecture of the society, indicating roles and possible interaction procedures. These procedures are also determined by the model chosen (network) and are informally described.

The institution underlying the society also imposes mechanisms for collaboration and certification. For instance, in the knowledge network a special kind of knowledge owner is responsible for the gathering and dissemination of information about a known, fixed list of subjects to knowledge seekers that subscribed to it. The institution must enforce the norm that such agents are required to provide all the information they are aware of. This determines a task for the monitors tracing this type of contracts of checking if information in all subjects in the list is indeed provided.

[1] This type of goods demands a complex matching mechanism, since matches are not at keyword level but require knowledge about relationships, processes etc. This imposes constraints to the task and communicative components of agents. This will not be discussed here.



`membership_application(X, gatekeeper):`
This is a negotiation between any agent and the gatekeeper of the society resulting in either an acceptance, that is X will become member of the society, or a rejection.
The role the agent will play is also determined in this scene.

`register(M, matchmaker):`
Knowledge owners or seekers can register their requests with the matchmaker, who will use this information in future matches

`request_partner(M, matchmaker):`
Knowledge owners or seekers request possible partners for an exchange.
Results in a possibly empty list of potential partners.

`negotiate_partnership(M, N):`
Owners and seekers check the viability of an exchange and determine conditions

`make_contract(M, N, notary):`
When an agreement is reached, partners register their commitments with the notary.

`appoint(notary, monitor):`
The notary appoints a monitor for a contract. It delegates agreed tasks to the monitor.
The monitor will keep track of contract status and will act when an undesired state is reached.

`apply_sanction(monitor, M):`
when a breech of contract occurs the monitor will contact the faulty party and apply the sanctions agreed upon (either described in the contract or standard in the institution).

Fig. 1. Fragment of the Knowledge Exchange Network architecture

6. Conclusions and Future Work

We have presented a framework for the design of agent societies based on the co-ordination structure of the domain that uses institutions to specify and enforce social norms and conventions. The framework takes the organisational perspective as starting point. We believe that one contribution of our research is that it describes the implications of the co-ordination model of the organisation for the architecture and design method of the agent society being developed. Although there are several agent-based software engineering methodologies (see, [8, 3, 16, 21]) these are often either too specific or too formal and not easily used and accepted. Our approach is to provide a generic frame that directly relates to the organisational perception of a problem. If needed, existing methodologies can be used for the development, modelling and formalisation of each step. We believe that our approach will contribute to the acceptance of multi-agent technology by organisations.

We also exposed the need for institutions in systems of autonomous agents that act according to their own goals and capabilities. Institutions enforce the global behaviour of the society and assure that the global goals of the society are met. Institutions play an important role to specify and manage the conventions of the agent society. One of the most important aspects is that they can make organisational goals and norms explicit and warrant their fulfilment by providing explicit facilitation roles and controlled interaction protocols. Different co-ordination models have different needs in terms of how institutions are specified. Feedback from the applications currently under development at Achmea will be used to improve the design methodology and the co-ordination frameworks used.

Important work that is left for the future is the formal description of both the co-ordination framework as well as the institutions. This will provide means for verifying properties of the institution. It will also enable agents that consider joining the society whether they are able and willing to conform to the specified conventions and interaction mechanisms.

References

1. Artikis, A., Kamara, L., Pitt, J.: Towards an Open Agent Society Model and Animation, Proceedings of the Agent-Based Simulation II workshop, Passau, (2001) 48-55
2. Bond, A., Gasser, L.: Readings in Distributed Artificial Intelligence. Morgan Kaufmann, (1988)
3. Brazier, F., Dunin-Keplicz, B., Jennings, N., Treur, J.: DESIRE: Modelling Multi-Agent Systems in a Compositional Formal Framework. In: Huhns, M., Singh M. (eds.): International Journal of Cooperative Information Systems, **6**(1) (1997)
4. Brazier, F., Jonker, C., Treur, J.: Compositional Design and Reuse of a Generic Agent Model. Applied Artificial Intelligence Journal, **14**, (2000) 491-538.
5. Castelfranchi, C.: Engineering Social Order, Omicini, A., Tolksdorf, R., Zambonelli, F., (Eds.) Engineering Societies in the Agents World, First International Workshop, ESAW 2000, Berlin, Germany, LNAI 1972, Springer-Verlag (2000) 1 – 19
6. Castillo, A., Kawaguchi, M., Paciorek, N., Wong, D.: Concordia™ as Enabling Technology for Cooperative Information Gathering. In: Proceedings of Japanese Society for Artificial Intelligence Conference, Tokyo, Japan (1998)
7. Dellarocas, C.: Contractual Agent Societies: Negotiated shared context and social control in open multi-agent systems. In: Proceedings of Workshop on Norms and Institutions in Multi-Agent Systems, Autonomous Agents-2000, Barcelona (2000)
8. DeLoach, S.: Multiagent Systems Engineering: A Methodology and Language for Designing Agent Systems. In: Proceedings of Workshop on Agent-Oriented Information Systems (AOIS'99) (1999)
9. Dignum, F.: Autonomous Agents with Norms. In AI and Law, (**7**), (1999) 69 – 79.
10. Dignum, F.: Agents, Markets, Institutions and Protocols. In Dignum, F., Sierra, C. (eds.): Agent Mediated Electronic Commerce, LNAI 1991, Springer-Verlag (2001) 98 – 114.
11. Dignum, V., Weigand, H., Xu L.: Agent Societies: Towards framework-based design. In: Wooldridge, M., Ciancarini P., Weiss, G. (Eds.): Proceedings of the 2nd Workshop on Agent-Oriented Software Engineering, Autonomous Agents, Montreal, (2001) 25-31.

12. Esteva, M., Padget, J., Sierra, C.: Formalizing a language for Institutions and Norms. Proceedings of the 8th International Workshop on Agent Theories, Architectures and Languages, ATAL-2001, Seattle, (2001)
13. Ferber, J., Gutknecht, O.: A meta-model for the analysis and design of organizations in multi-agent systems. In: Proceedings of the Third International Conference on Multi-Agent Systems (ICMAS'98), IEEE Computer Society, (1998)
14. Frei, C., Faltings, B.: A Dynamic Hierarchy of Intelligent Agents for Network Management. In: Proceedings of 2nd International Workshop on Intelligent Agents for Telecommunications Applications (IATA'98), Paris, France (1998) 1-16
15. Jonker, C., Klusch, M., Treur, J.: Design of Collaborative Information Agents. In: Klusch M., Kerschberg, L. (eds.): Cooperative Information Agents IV. LNAI 1860, Springer-Verlag (2000) 262 – 283
16. Omicini, A.: SODA: Societies and Infrastructures in the Analysis and Design of Agent-based Systems. In: Ciancarini P., Wooldridge, M. (eds.): Agent-Oriented Software Engineering, LNCS 1957, Springer-Verlag (2001)
17. Ossowski, S.: Co-ordination in Artificial Agent Societies, LNAI 1535, Springer (1998)
18. Preece, A., Hui K., Gray, P.: KRAFT: Supporting Virtual Organisations through Knowledge Fusion. In: Finin T., Grosof B. (Eds): Artificial Intelligence for Electronic Commerce: Papers from the AAAI-99 Workshop, AAAI Press, (1999) 33-38.
19. Rocha, A.P., Oliveira, E.: An Electronic Market Architecture for the formation of Virtual Enterprises.Proceedings of PRO-VE'99 IFIP/PRODNET Conference on Infrastructures for Industrial Virtual Enterprises, Porto, October (1999)
20. Tsvetovat, M., Sycara, K., Chen, Y., Ying, J.: Customer Coalition in Electronic Markets. In: Dignum, F., Cortés, U. (Eds.): AMEC III, LNAI 2003 (2001) 121-138
21. Wooldridge, M., Jennings, N., Kinny, D.: The Gaia Methodology for Agent-Orient Analysis and Design. Autonomous Agents and Multi-Agent Systems, **3**(3) (2000)
22. Zambonelli, F., Jennings, N., Omicini, A., Wooldridge, M.: Agent-Oriented Software Engineering for Internet Applications. In: A. Omicini, Zambonelli, F., Klusch, M., Tolkdorf, R. (eds.): Coordination of Internet Agents: Models, Technologies, and Applications. Springer-Verlag (2001) 326 - 346
23. Zambonelli F., Jennings, N., Wooldridge, M.: Organisational Abstractions for the Analysis and Design of Multi-Agent Systems. In: Ciancarini P., Wooldridge, M. (eds.): Agent-Oriented Software Engineering, LNCS 1957, Springer-Verlag (2001)

Argumentation as Distributed Belief Revision: Conflict Resolution in Decentralised Co-operative Multi-agent Systems

Benedita Malheiro[1,2] and Eugénio Oliveira[1,3]

[1] LIACC - Artificial Intelligence and Computer Science Laboratory
http://www.ncc.up.pt/liacc/index.html
[2] ISEP, DEE, Rua Dr. António Bernardino de Almeida, 431, 4200-072 Porto, Portugal
bene@dee.isep.ipp.pt
[3] FEUP, DEEC, Rua Dr. Roberto Frias, 4200-465 Porto, Portugal
eco@fe.up.pt

Abstract. Decentralised co-operative multi-agent systems are computational systems where conflicts are frequent due to the nature of the represented knowledge. Negotiation methodologies, in this case argumentation based negotiation methodologies, were developed and applied to solve unforeseeable and, therefore, unavoidable conflicts.
The supporting computational model is a distributed belief revision system where argumentation plays the decisive role of revision. The distributed belief revision system detects, isolates and solves, whenever possible, the identified conflicts. The detection and isolation of the conflicts is automatically performed by the distributed consistency mechanism and the resolution of the conflict, or belief revision, is achieved via argumentation.
We propose and describe two argumentation protocols intended to solve different types of identified information conflicts: context dependent and context independent conflicts. While the protocol for context dependent conflicts generates new consensual alternatives, the latter chooses to adopt the soundest, strongest argument presented. The paper shows the suitability of using argumentation as a distributed decentralised belief revision protocol to solve unavoidable conflicts.

1 Introduction

Co-operative, decentralised multi-agent systems are computational systems composed of autonomous agents capable of solving distributed complex problems, which they are unable to do individually. Each agent plays the role of an expert in a specific system knowledge sub-domain and contributes within its sphere of competence for the achievement of the overall goal of the system. Since these systems have no central control, the agents are fully responsible for the information they share or the actions they take and, therefore, have to rely on appropriate methodologies for co-operation.

Co-operative interactions generate elaborated inter-agent knowledge dependency networks which, in our case, are based on the assumption that trustful, benevolent agents convey only correct information and results. However, this assumption does not guarantee the overall consistency of the system's knowledge since agents are resource bounded entities that have a partial understanding of the global problem and suffer from incomplete knowledge. Consequently, conflicts regarding the beliefs of the different agents are frequently declared. These disparate perspectives can either be reconcilable or incompatible, falling, respectively, into the categories of negative and positive conflicts [14]. In particular, this paper is concerned with solving negative conflicts that occur when agents hold contradictory beliefs.

Negotiation based conflict resolution protocols, particularly argumentation protocols, are well suited for autonomous co-operative agents. In argumentation based negotiation, or simply argumentation, the agents generate and exchange arguments to support their findings or conclusions, evaluate them and agree to accept the soundest arguments presented or the first consensual alternative proposed.

In this paper, argumentation is regarded a distributed revision protocol and, thus, is a functionality of a distributed decentralised belief revision system. The autonomous agents are modelled as individual reason maintenance systems where beliefs are always represented with their supporting justifications. A belief's justification constitutes a full argument in favour of the belief and, in our case, is composed by the set of elementary beliefs (called foundations) that support the belief. So, when an agent shares a belief with others, it sends the belief together with its supporting arguments. The receiver agents, with this additional information, verify if the arrived belief is consistent with the already represented beliefs and, in case of conflict, immediately identify and isolate the conflicting set of beliefs in order to preserve consistency. However, consistency maintenance alone does not solve conflicts. In order to (try to) solve the detected conflicts negotiation-based strategies (in our case argumentation) need to be applied.

We propose and describe two argumentation protocols intended to solve different types of identified information conflicts: context dependent and context independent conflicts. While the protocol for context dependent conflicts generates new consensual alternatives, the latter chooses to adopt the soundest, strongest argument presented. The strategies implemented are the decision functions used to choose the soundest argument or to provide new conflict-free alternatives.

The objective of this paper is to show the appropriateness of argumentation as a distributed belief revision protocol for decentralised co-operative conflict solving by describing how this extension was implemented and by presenting the developed argumentation strategies.

After this introductory section we briefly present our motivation and approach and describe the argumentation based conflict solving protocol developed. We start by presenting the protocol for solving Context Independent Conflicts and then follow with the protocol for solving the Context Dependent Conflicts.

In the fifth section we discuss our work by comparing it with related work, and, in the last section we draw our conclusions.

2 Motivation and Approach

The purpose of the majority of the negotiation protocols found in the literature (for example [10], [15], [17] or [16]) is to avoid the declaration of potential future conflicts. A typical case is when two agents detect that their future actions, goals or plans are conflicting and use negotiation to find conflict-free alternatives. The work presented in this paper, rather than avoiding the declaration of conflicts, is focused on the actual resolution of conflicts that cannot be predicted beforehand and, therefore, must be identified, isolated and, whenever possible, solved.

The main motivation of this work came from the proposed problem domain: the determination of appropriate locations for new project developments. This real world domain consists of multiple public evaluation agencies that typically reach verdicts independently, i.e., without realising that they share knowledge (legislation) and that their judgements and recommendations are interdependent. This type of procedure results in long and costly (re-)submission, evaluation, alteration cycles both for the public agencies as well as for project developers. The goal was to design an intelligent decision-making tool capable of finding adequate project locations while correcting this undesirable real world behaviour.

The approach followed was modular and incremental. First, the identified features of the problem domain suggested the design of a co-operative decentralised multi-agent system where distributed volatile information could be adequately represented, used and maintained. The individual agents were remotely inspired by the ARCHON architecture [18], and are structured in two main layers: the intelligent system layer and the cooperation layer. The intelligent system layer is modelled as a reason maintenance system which includes an Assumption based Truth Maintenance Systems (ATMS) [2]. A reason maintenance system can represent and reason with dynamic data by using beliefs since beliefs, unlike facts, are represented with their supporting justifications. The agents knowledge bases (composed of beliefs and facts) are updated according to the most recent findings and their consistency is preserved by the automatic detection and isolation of conflicts. The cooperation layer includes a self model, an acquaintances model and a communication module. The description of the architectural and functional details of the system can be found in [11]. Another possible agent architecture could be a BDI-like architecture [7]. The BDI agents are considered rational agents that have certain mental attitudes of Belief, Desire and Intention (BDI), representing, respectively, the information, the motivational and the deliberative states of the agent.

On top of these functionalities, we added an argumentation extension responsible for solving the identified information conflicts. The autonomous agents engage in co-operative interactions by sharing beliefs and their justifications, maintain and update their internal beliefs, and are responsible for communicating relevant changes (revisions included) to their counterparts. The resulting distributed belief revision system is composed of an initial consistency preservation stage followed by a final argumentation-based conflict resolution phase. This view of belief revision as a truth maintenance process followed by a selection mechanism of one (or more) preferred solutions was also proposed by [3].

We developed two distributed belief revision protocols to solve:

– *Context Independent Conflicts*, which occur when a distributed belief is, simultaneously, *believed* by some agents and *disbelieved* by others;
– *Context Dependent Conflicts*, which occur when the agents detect inconsistent distinct beliefs.

In the first case, the goal is to choose the most appropriate belief status to adopt, and, in the latter case, to find consensual alternatives to support the affected beliefs. The proposed conflict resolution protocols, although distinct from the typical negotiation based conflict resolution protocols that perform a distributed search trough the space of possible solutions [8], have identical goals: both try to reach acceptable agreements to all the parties involved by:

– choosing between conflicting views, by comparing the reasons behind each stance and choosing the strongest view;
– building a new consensual view, by searching for fully acceptable alternative foundations for the disputed information.

The problem domain was then modelled, using this architecture, as a cooperative multi-agent composed of autonomous agents with belief revision capabilities. The resulting distributed belief revision system contains two types of functional agents: decision-making agents and information providing agents. While the information providing agents classify and provide diverse thematic information for a given geographic region (usually include geographical information systems), the role of the decision-making agents is to find locations that satisfy the requirements of the submitted projects and also comply with the legislation applicable to the geographical region under consideration.

The distributed belief revision activity plays a key role in this platform since it allows the autonomous agents to take into account the existing knowledge dependencies, to detect any information conflicts and, whenever possible, solve the distributed conflicts. Argumentation is particularly well suited for this domain because the system needs to produce a final verdict which can be justified and that results from a global consensus. Suppose, for example, that the system is trying to find a location for a new farm development and a conflict between $Agent_1$ and $Agent_2$ occurs: $Agent_1$ believes that *"sandstone is appropriate for agriculture"* and $Agent_2$ disbelieves it. This conflict, which is domain independent, is addressed via argumentation: the agents argue in favour of their own perspective by calculating and exchanging the respective credibility and agree to adopt the most credible perspective. Consider that another conflict is then detected between $Agent_1$ and $Agent_2$: $Agent_1$ believes that *"irrigated land is appropriate for agriculture"* and $Agent_2$ believes that *"water resources must be* *preserved"*. This conflict, which is domain dependent, is also addressed via argumentation: the agents try to find new consensual alternatives regarding the concepts in conflict. $Agent_2$ informs $Agent_1$ that it has no alternatives. $Agent_1$ finds as a candidate that *"range land is appropriate for agriculture"*. This alternative is evaluated and since it is consensual it is assumed by $Agent_1$.

The resulting system, named *Distributed Project Location Multi-Agent Test bed* (DIPLoMAT), is described in [12]. The DIPLoMAT system determines appropriate locations for the specified projects and, simultaneously, allows valuable what/if analysis. This ability, which is supported by distributed belief revision, provides immediate answers to questions like What if a specific legislation rule is altered?, What if a particular project requirement is changed?, and so forth.

Before continuing with our paper we wish to present some definitions:

1. Beliefs are first order logic sentences;
2. Beliefs result from perception, communication, assumption or inference;
3. Foundational beliefs or foundations are self-supported elementary beliefs which follow directly from perception or assumption;
4. A belief's justification is a minimal set of foundations from which it depends.
5. A belief ϕ of agent Ag (also referred as agent Ag view of ϕ) is represented by a tuple (also called an ATMS node) $< \phi_{Ag}, \mathcal{E}(\phi_{Ag}), \mathcal{F}(\phi_{Ag}), Ag >$, where ϕ_{Ag} identifies the belief, $\mathcal{E}(\phi_{Ag})$ specifies the belief's endorsement (observed, assumed, communicated or inferred), $\mathcal{F}(\phi_{Ag})$ contains the belief's supporting justifications, and Ag identifies the belief's source agent. The belief status is established according to the $\mathcal{F}(\phi_{Ag})$: (i) *Believed*, if $\mathcal{F}(\phi_{Ag}) \neq \{\emptyset\}$; (ii) *Disbelieved*, if $\mathcal{F}(\phi_{Ag}) = \{\emptyset\}$;
6. A distributed belief ϕ is composed by all existing beliefs regarding ϕ, i.e., includes every agent view of ϕ. For example, $< \phi_1, \mathcal{E}(\phi_1), \mathcal{F}(\phi_1), Ag_1 >$, ..., $< \phi_n, \mathcal{E}(\phi_n), \mathcal{F}(\phi_n), Ag_n >$.

3 Context Independent Conflicts

The Context Independent Conflicts result from the assignment, by different agents, of contradictory belief statuses to the same belief. Every agent maintains not only its individual beliefs but also participates on the maintenance of the distributed beliefs that include its own views. While the responsibility for individual beliefs relies on each source agent, the responsibility for distributed beliefs is shared by the group of agents involved. Three elementary processing criteria for the accommodation of distributed beliefs were implemented:

– *The CONsensus (CON) criterion* - The distributed belief will be *Believed*, if all the perspectives of the different agents involved are believed, or *Disbelieved*, otherwise;
– *The MAJority (MAJ) criterion* - The distributed belief will be *Believed*, as long as the majority of the perspectives of the different agents involved is believed, and *Disbelieved*, otherwise;

– *The At Least One (ALO) criterion* - The distributed belief will be *Believed* as long as at least one of the perspectives of the different agents involved is believed, and *Disbelieved*, otherwise.

By default, i.e., when the distributed belief is consensual, the CON criterion is applied, otherwise the distributed belief results from the resolution of the detected Context Independent Conflict. The methodology we are proposing for solving the Context Independent Conflicts is organized in two steps: first, it establishes the desired outcome of the social conflict episode, and then, it applies the processing criterion that solves the episode accordingly. During the first stage, the conflict episode is analysed to establish the most reasonable outcome. The necessary knowledge is extracted from data dependent features like the agents' reliability [4] and the endorsements of the beliefs [5], allowing the selection, at runtime, of the most adequate processing criterion. In particular, in our work the reliability of an agent is domain dependent - an agent can be more reliable in some domains than in others. The dynamic selection of the processing criterion is based on assessment of the credibility values associated with each belief status. Two credibility assessment procedures were designed:

The Foundations Origin based Procedure (FOR) - where the credibility of the conflicting perspectives is determined based on the strength of the foundations endorsements (observed foundations are stronger than assumed foundations) and on the domain reliability of the source agents; and
The Reliability based Procedure (REL) - where the credibility of the conflicting views is based on the reliability of the foundations source agents.

The methodology for solving Context Independent Conflicts starts by applying the FOR procedure. If the FOR procedure is able to determine the most credible belief status, then the selected processing criterion is applied and the episode is solved. However, if the result of the application of the FOR procedure is a draw between the conflicting perspectives, the Context Independent conflict solving methodology proceeds with the application of the REL procedure. If the REL procedure is able to establish the most credible belief status, then the selected processing criterion is applied and the episode is solved.

The sequential application of these procedures is ordered by the amount of knowledge used to establish the resulting belief status. It starts with the FOR procedure which calculates the credibility of the conflicting perspectives based on the strength of the endorsements and on the domain reliability of the sources (agents) of the foundations. It follows with the REL procedure which computes the credibility of the conflicting beliefs solely based on the domain reliability of the sources of the foundations. We will now describe in detail the procedures for solving Context Independent Conflicts mentioned above.

3.1 The Foundations ORigin Based Procedure (FOR)

Generic beliefs are supported by sets of foundational beliefs which follow directly from observation, assumption or external communication. Since communicated beliefs also resulted from some process of observation, assumption or communication of other agents, foundations are, ultimately, composed of just observed or assumed beliefs. Galliers [5] refers to a belief's process of origin as the belief's endorsement.

When a Context Independent Conflict involving a distributed belief ϕ occurs the FOR procedure is invoked and the credibility values for each one of the conflicting views regarding ϕ is computed according to the following formulas:

$$\mathcal{C}(\phi, Bel) = \textstyle\sum_{Ag=i}^{k} \mathcal{C}(\phi_{Ag}, Bel)/N$$
$$\mathcal{C}(\phi, Dis) = \textstyle\sum_{Ag=i}^{k} \mathcal{C}(\phi_{Ag}, Dis)/N$$

where N is the number of agents involved in the conflict. The values of the credibility attached to each agent view are equal to the average of the credibility values of their respective sets of foundations. The credibility of a generic foundation of ϕ - for example, $< \alpha_{Ag}, \mathcal{E}(\alpha_{Ag}), \mathcal{F}(\alpha_{Ag}), Ag >$ which belongs to domain D - depends on the reliability of the foundation's source agent in the specified domain ($\mathcal{R}(D, Ag)$), on the foundation's endorsement ($\mathcal{E}(\alpha_{Ag})$) and on its support set ($\mathcal{F}(\alpha_{Ag})$):

$$\mathcal{C}(\alpha_{Ag}, Bel) = 1 \times \mathcal{R}(D, Ag), \text{ if } \mathcal{F}(\alpha_{Ag}) \neq \{\emptyset\} \text{ and } \mathcal{E}(\alpha_{Ag}) = obs;$$
$$\mathcal{C}(\alpha_{Ag}, Bel) = 1/2 \times \mathcal{R}(D, Ag), \text{ if } \mathcal{F}(\alpha_{Ag}) \neq \{\emptyset\} \text{ and } \mathcal{E}(\alpha_{Ag}) = ass;$$
$$\mathcal{C}(\alpha_{Ag}, Bel) = 0 \text{ if } \mathcal{F}(\alpha_{Ag}) = \{\emptyset\};$$
$$\mathcal{C}(\alpha_{Ag}, Dis) = 1 \times \mathcal{R}(D, Ag), \text{ if } \mathcal{F}(\alpha_{Ag}) = \{\emptyset\} \text{ and } \mathcal{E}(\alpha_{Ag}) = obs;$$
$$\mathcal{C}(\alpha_{Ag}, Dis) = 1/2 \times \mathcal{R}(D, Ag), \text{ if } \mathcal{F}(\alpha_{Ag}) = \{\emptyset\} \text{ and } \mathcal{E}(\alpha_{Ag}) = ass;$$
$$\mathcal{C}(\alpha_{Ag}, Dis) = 0 \text{ if } \mathcal{F}(\alpha_{Ag}) \neq \{\emptyset\};$$

Within this procedure, the credibility granted, *a priori*, to observed foundations and assumed foundations was, respectively, 1 and 1/2. The credibility of each foundation is also affected by the reliability of the origin agent (source agent) for the domain under consideration. As a result, each perspective is assigned a credibility value equal to the average of the credibility values of its support foundations. The credibility of any perspective has, then, a value between 0 and 1. A credibility value of 1 means that perspective is 100% credible (it solely depends on observed foundations generated by agents fully reliable on the data domain), whereas a credibility value of 0 means that no credibility whatsoever is associated with the perspective. Semantically, the 1 and 1/2 values granted to observed and assumed foundations have the following meaning: evidences corroborated by perception are 100% credible, whereas assumptions have a 50% chance of being confirmed or contradicted through perception.

The FOR procedure calculates the credibility attached to the each one of the conflicting belief statuses (*Believed* and *Disbelieved*), and chooses the multiple perspective processing criterion whose outcome results in most credible belief status. If the most credible belief status is: (i) *Disbelieved*, then the CON criterion is applied to the episode of the conflict; (ii) *Believed*, then, if the majority of the perspectives are in favour of believing in the belief the MAJ criterion is applied; else the ALO criterion is applied to the episode of the conflict. The agents' reliability on the domain under consideration is affected by the outcome of the

Context Independent Conflict episodes processed so far. An episode winning agent increases its reliability in the specified domain, while an episode loosing agent decreases its reliability in the specified domain. At launch time, the agents are assigned a reliability value of 1 to every knowledge domain, which, during runtime, may vary between 0 and 1. If the agent's view won the conflict episode of domain D then

> $\mathcal{R}(D, Ag) = \mathcal{R}(D, Ag) \times (1 + N_w/N) \times f_{norm}$, where N_w, N and f_{norm} represent, respectively, the number of agents who won the episode, the total number of agents involved, and a normalization factor needed to keep the resulting values within the interval [0,1];

If agent Ag view lost a Context Independent Conflict episode of domain D then

> $\mathcal{R}(D, Ag) = \mathcal{R}(D, Ag) \times (1 - N_l/N) \times f_{norm}$, where N_l, N and f_{norm} represent, respectively, the number of agents who lost the episode, the total number of agents involved in the conflict episode and a normalization factor needed to keep the resulting values within the interval [0,1].

A domain reliability value of 1 means that the agent has been fully reliable, and a value near 0 means that the agent has been less than reliable.

Example Suppose a multi-agent system composed of three agents, $Agent_1$, $Agent_2$ and $Agent_3$, with the following knowledge bases:

$Agent_1$ has observed $\alpha(a)$, assumed $\beta(a)$ and has two knowledge production rules[1], $r_{1,1} : \alpha(X) \wedge \beta(X) \rightarrow \phi(X)$ and $r_{1,2} : \delta(X) \wedge \phi(X) \rightarrow \psi(X)$:
$< \alpha_1(a), obs, \{\{\alpha_1(a)\}\}, Agent_1 >$;
$< \beta_1(a), ass, \{\{\beta_1(a)\}\}, Agent_1 >$;
$< r_{1,1}, obs, \{\{r_{1,1}\}\}, Agent_1 >$;
$< r_{1,2}, obs, \{\{r_{1,2}\}\}, Agent_1 >$;
$Agent_2$ has observed $\alpha(a)$, assumed $\delta(a)$ and has one knowledge production rule, $r_{2,1} : \alpha(X) \wedge \delta(X) \rightarrow \phi(X)$:
$< \alpha_2(a), obs, \{\{\alpha_2(a)\}\}, Agent_2 >$;
$< \delta_2(a), ass, \{\{\delta_2(a)\}\}, Agent_2 >$;
$< r_{2,1}, obs, \{\{r_{2,1}\}\}, Agent_2 >$;
$Agent_3$ has observed $\phi(a)$:
$< \phi_3(a), obs, \{\{\phi_3(a)\}\}, Agent_3 >$.

$Agent_1$ is interested in receiving information regarding $\alpha(X)$, $\beta(X)$, $\delta(X)$, $\phi(X)$ and $\psi(X)$, $Agent_2$ is interested in $\alpha(X)$, $\delta(X)$ and $\phi(X)$, and $Agent_3$ is interested in any information on $\phi(X)$. The agents after sharing their results (according to their expressed interests) end up with the following new beliefs:

$Agent_1$ has received $\alpha_2(a)$, $\delta_2(a)$, $r_{2,1}$, $\phi_3(a)$, and $\phi_2(a)$, and has inferred $\phi_1(a)$ and $\psi_1(a)$:

[1] rules are also represented as beliefs which may be activated or inhibited (believed/disbelieved).

$< \alpha_2(a), com, \{\{\alpha_2(a)\}\}, Agent_2 >$;
$< \delta_2(a), com, \{\{\delta_2(a)\}\}, Agent_2 >$;
$< r_{2,1}, com, \{\{r_{2,1}\}\}, Agent_2 >$;
$< \phi_3(a), com, \{\{\phi_3(a)\}\}, Agent_3 >$;
$< \phi_1(a), inf, \{\{\alpha_1(a), \alpha_2(a), \beta_2(a), r_{1,1}\}\}, Agent_1 >$;
$< \phi_2(a), com, \{\{\alpha_1(a), \alpha_2(a), \delta_2(a), r_{2,1}\}\}, Agent_2 >$;
$< \psi_1(a), inf, \{\{\alpha_1(a), \alpha_2(a), \beta_1(a), \delta_2(a), r_{1,1}, r_{2,1}, \phi_3(a), r_{1,2}\}\}, Agent_1 >$;

$Agent_2$ has received $\alpha_1(a)$, $\beta_1(a)$, $\phi_3(a)$, $r_{1,1}$, and $\phi_1(a)$, and has inferred $\phi_2(a)$:

$< \alpha_1(a), com, \{\{\alpha_1(a)\}\}, Agent_1 >$;
$< \beta_1(a), com, \{\{\beta_1(a)\}\}, Agent_1 >$;
$< r_{1,1}, com, \{\{r_{1,1}\}\}, Agent_1 >$;
$< \phi_3(a), com, \{\{\phi_3(a)\}\}, Agent_3 >$;
$< \phi_2(a), inf, \{\{\alpha_1(a), \alpha_2(a), \delta_2(a), r_{2,1}\}\}, Agent_2 >$;
$< \phi_1(a), com, \{\{\alpha_1(a), \alpha_2(a), \beta_1(a), r_{1,1}\}\}, Agent_1 >$;

$Agent_3$ has received $\alpha_1(a)$, $\alpha_2(a)$, $\beta_1(a)$, $\delta_2(a)$, $r_{1,1}$, $r_{2,1}$, $\phi_1(a)$, $\phi_2(a)$ $r_{1,1}$, $\phi_1(a)$, and has inferred $\phi_2(a)$:

$< \alpha_1(a), com, \{\{\alpha_1(a)\}\}, Agent_1 >$;
$< \alpha_2(a), com, \{\{\alpha_2(a)\}\}, Agent_2 >$;
$< \beta_1(a), com, \{\{\beta_1(a)\}\}, Agent_1 >$;
$< \delta_2(a), com, \{\{\delta_2(a)\}\}, Agent_2 >$;
$< r_{1,1}, com, \{\{r_{1,1}\}\}, Agent_1 >$;
$< r_{2,1}, com, \{\{r_{2,1}\}\}, Agent_2 >$;
$< \phi_1(a), com, \{\{\alpha_1(a), \alpha_2(a), \beta_1(a), r_{1,1}\}\}, Agent_1 >$;
$< \phi_2(a), com, \{\{\alpha_1(a), \alpha_2(a), \delta_2(a), r_{2,1}\}\}, Agent_2 >$;

At some point, $Agent_3$ realizes, via observation, that $\phi(a)$ is no longer believed, i.e., $< \phi_3(a), obs, \{\emptyset\}, Agent_3 >$. A first episode of a Context Independent Conflict regarding $\phi(a)$ is declared: $\phi_1(a)$ and $\phi_2(a)$ are believed while $\phi_3(a)$ is disbelieved. In the case of our conflict, the credibility values assigned to the believed status is obtained through the following expressions:

$$\begin{aligned}
&\mathcal{C}(\phi_1(a), Bel) = (\mathcal{C}(\alpha_1(a), Bel) + \mathcal{C}(\alpha_2(a), Bel) + \mathcal{C}(\beta_1(a), Bel) + \mathcal{C}(r_{1,1}, Bel))/4 \\
&\mathcal{C}(\phi_2(a), Bel) = (\mathcal{C}(\alpha_1(a), Bel) + \mathcal{C}(\alpha_2(a), Bel) + \mathcal{C}(\delta_2(a), Bel) + \mathcal{C}(r_{2,1}, Bel))/4 \\
&\mathcal{C}(\phi_3(a), Bel) = \mathcal{C}(\phi_3(a), Bel)
\end{aligned}$$

As the default reliability values assigned to every agent domain was 1, the credibility values associated to the conflicting perspectives are:

$$\begin{array}{lll}
\mathcal{C}(\phi_1(a), Bel) = (1 + 1 + 1/2 + 1)/4 & \quad and \quad & \mathcal{C}(\phi_1(a), Dis) = 0 \\
\mathcal{C}(\phi_2(a), Bel) = (1 + 1 + 1/2 + 1)/4 & \quad and \quad & \mathcal{C}(\phi_2(a), Dis) = 0 \\
\mathcal{C}(\phi_3(a), Bel) = 0 & \quad and \quad & \mathcal{C}(\phi_3(a), Dis) = 1
\end{array}$$

resulting in

$$\mathcal{C}(\phi(a), Bel) = 7/12 \qquad and \qquad \mathcal{C}(\phi(a), Dis) = 1/3.$$

Since $\mathcal{C}(\phi(a), Bel) > \mathcal{C}(\phi(a), Dis)$, the multi-agent system decides to believe in $\phi(a)$. The first episode of the conflict regarding $\phi(a)$ was successfully solved through the application of the FOR procedure, and the processing criterion that must be applied to generate the adequate social outcome of this conflict episode is the MAJ criterion. Finally, the conflict domain reliability values of the agents involved in the conflict episode are updated accordingly. So $Agent_1$, $Agent_2$ and $Agent_3$ updated credibility values for the domain under consideration (D) are:

$\mathcal{R}(D, Agent_1) = 1 \times (1 + 2/3)/(1 + 2/3)$, i.e., $\mathcal{R}(D, Agent_1) = 1$,
$\mathcal{R}(D, Agent_2) = 1 \times (1 + 2/3)/(1 + 2/3)$, i.e., $\mathcal{R}(D, Agent_2) = 1$, and
$\mathcal{R}(D, Agent_3) = 1 \times (1 - 1/3)/(1 + 2/3)$, i.e., $\mathcal{R}(D, Agent_3) = 6/15$.

3.2 The RELiability Based Procedure (REL)

The REL procedure assigns each conflicting view a credibility value equal to the average of the reliability values of the source agents of its foundations. The credibility associated with the different perspectives that contribute to each belief status are added and the REL procedure chooses the processing criterion whose outcome results in adopting the most credible belief status. If the most credible belief status is: (i) *Disbelieved*, then the CON criterion is applied to the episode of the conflict; (ii) *Believed*, then, if the majority of the perspectives are in favour of believing in the belief the MAJ criterion is applied, else the ALO criterion is applied to the episode of the conflict. The reliability of the agents in the domain conflict is also affected by the outcome of the conflict episodes solved so far. Episode winning agents increase their reliability in the conflict domain, while episode loosing agents decrease their reliability in the conflict domain (see previous sub-section).

4 Context Dependent Conflicts

The detection of contradictory distinct beliefs naturally unchains the reason maintenance mechanism. However, in the case of the Context Dependent Conflicts, consistency preservation is not immediately followed the application of conflict resolution strategies. The agents postpone resolution in an effort to preserve their current beliefs since the resolution of Context Dependent Conflicts implies abandoning the foundations of the conflicting beliefs and generating alternative consensual foundational beliefs. Resolution is delayed until the moment when the system concludes that it cannot fulfil its goals unless it tries to solve the latent conflicts. To solve this type of conflicts the agents need to know how to provide alternative support to the invalidated conclusions. This search for "next best" solutions is a relaxation mechanism called *Preference ORder based procedure (POR)*.

Each knowledge domain has lists of ordered candidate attribute values for some domain concepts. These lists contain sets of possible instances ordered by preference (the first element is the best candidate, the second element is the second best, and so forth) for the attribute values of the specified concepts. The preference order values are affected by the proposing agents' reliability in the domain under consideration. In the case of a foundational belief maintained by a single agent, the process is just a next best candidate retrieval operation. In the case of a distributed foundation, a consensual candidate has to be found. If the gathered proposals are:

– *Identical*, then the new foundation has been found;
– *Different*, then the agents that proposed higher preference order candidates generate new next best proposals, until they run out of candidates or a consensual candidate is found.

The alternative foundations found through this strategy are then assumed by the system. The credibility measure of the resulting new foundations is a function of the lowest preference order candidate used and of the involved agents' reliability. The agents' reliability values in the domain under consideration are not affected by the resolution of Context Dependent Conflicts.

5 Related Work

Although the work presented in this paper was inspired by many previous contributions from research fields such as belief revision, argumentation based negotiation and multi-agent systems, the approach followed is, as far as we know, novel.

Authors like [16], [15] or [17] propose argumentation based negotiation protocols for solving conflicts in multi-agent systems. However, they do not address the problem of how to solve already declared conflicts, but concentrate in avoiding the possible declaration of future conflicts. Several approaches to argumentation have been proposed: the definition of logics [10], detailed agent mental states formalisation or the construction and structure of valid arguments [17]. Nevertheless, we believe that argumentation as distributed belief revision has additional advantages over other argumentation protocols implementations. This claim is supported by the fact that in our computational model the supporting arguments are automatically shared during co-operation. For instance, when a context independent conflict occurs, since the arguments in favour of the conflicting beliefs have already been exchanged, the FOR and REL conflict resolution strategies are immediately applied speeding up the process. Furthermore, argumentation as distributed belief revision allows pro-active context independent conflict resolution, since when this type of conflict occurs the agents accept the social conflict outcome while retaining their own individual perspectives. As a result, the smallest change will trigger the re-evaluation of the conflict status.

The implemented autonomous belief revision strategies were also build on the proposals of other authors. The idea of using endorsements for determining preferred revisions was first proposed by Cohen [1] and, later, by Galliers in [5]. However, while Galliers proposes several types of endorsements according to the process of origin of the foundational beliefs (communicated, given, assumed, etc.), we only consider two kinds of endorsement: perception or assumption. We base this decision on the fact that beliefs result from some process of observation, assumption or external communication, and since communicated perspectives also resulted from some process of observation, assumption or communication of other agents, ultimately, the foundations set of any belief is solely made of observed and assumed beliefs.

Similarly, Gaspar [6] determines the preferred belief revisions according to a belief change function which is based on a set of basic principles and heuristic criteria (sincerity of the sending agent, confidence and credulity of the receiving agent) that allow the agents to establish preferences between contradictory beliefs. Beliefs are associated to information topics and different credibility values are assigned to the agents in these topics. More recently, Dragoni and Giorgini [4] proposed the use of a belief function formalism to establish the credibility of the beliefs involved in conflicts according to the reliability of the source of belief and to the involved beliefs credibility. Each agent is assigned a global reliability value which is updated by the belief function formalism after each conflict. Our agents, like Gaspar's, have domain dependent reliability values because we believe that agents tend to be more reliable in some domains than others and that the assignment of a global agent reliability (like Dragoni and Giorgini do) would mask these different expertise levels. We also guarantee, unlike Dragoni and Giorgini, that communicated foundations are only affected by the reliability of the foundation source agent, and not by the reliability of agent that communicated the foundation (which may be different).

Finally, a quite interesting approach is presented by [9] where argumentation is also regarded as an extension to an existing distributed computational model. The authors regard argumentation as constraint propagation and use, consequently, a distributed constraint satisfaction computational model.

6 Conclusion

This research was motivated by the need to design a co-operative decentralised multi-agent system where dynamic, incomplete data could be adequately represented, used and maintained. The idea of using argumentation as a distributed belief revision protocol to solve information conflicts was a natural consequence of the adopted approach since the autonomous agents were modelled as individual reason maintenance systems. An important advantage of this approach lies on the fact that arguments are automatically exchanged during co-operation and are already available when conflicts occur. As a result, the declaration of conflicts results in the immediate application of the implemented evaluation strategies.

The belief revision strategies designed for the resolution of the identified types of negative conflicts are based on data dependent features or explicit knowledge rather than belief semantics. The data features used to solve conflicting beliefs have been previously proposed by other authors (source agent reliability by Dragoni and Giorgini [4], endorsements by Galliers [5], and specification of preferences between beliefs by Gaspar [6]). However, we believe that we are combining them in a novel and more efficient manner.

These argumentation protocols, which are implemented in the DIPLoMAT system, are under test and evaluation. They attempt to solve the detected conflicting beliefs but cannot, beforehand, guarantee whether their effort will be successful or not.

References

1. Cohen, P. R.; Heuristic Reasoning about Uncertainty: an Artificial Intelligence Approach. Pitman, Boston (1985).
2. de Kleer, J.; An Assumption-Based Truth Maintenance System. Artificial Intelligence 28(2):127-224 (1986).
3. Dragoni, A. F. Giorgini, P. and Puliti, P.; Distributed Belief Revision versus Distributed Truth Maintenance. In Proceedings of the Sixth IEEE Conference on Tools with Artificial Intelligence, IEEE Computer Press (1994).
4. Dragoni, A. F. and Giorgini, P.; Belief Revision through the Belief Function Formalism in a Multi-Agent Environment. Third International Workshop on Agent Theories, Architectures and Languages, LNAI N. 1193, Springer-Verlag, (1997).
5. Galliers, R; Autonomous belief revision and communication, Gärdenfors, P. ed. Belief Revision. Cambridge Tracts in Theoretical Computer Science, Cambridge University Press, N. 29 (1992).
6. Gaspar, G.; Communication and Belief Changes in a Society of Agents: Towards a Formal Model of an Autonomous Agent, Demazeau, Y. and Müller, J. P. eds. Decentralized A. I., Vol. 2, North-Holland Elsevier Science Publishers (1991).
7. Kinny, D. N. and Georgeff M. P.; Commitment and effectiveness of situated agents. In Proceedings of the Twelfth International Joint Conference on Artificial Intelligence (IJCAI'91), Sydney, Australia (1991).
8. Jennings, N. R., Parsons, S., Noriega, P. and Sierra, C.; On Argumentation-Based Negotiation. In Proceedings of the International Workshop on Multi-Agent Systems, Boston, USA (1998).
9. Jung, H., Tambe, T., Parsons, S., and Kulkarni, S.; Argumentation as Distributed Constraint Satisfaction: Applications and Results. In Proceedings of the International Conference on Autonomous Agents (Agents'01), Montreal, Quebec, Canada (2001).
10. Kraus, S., Sycara, K., and Evenchik, A.; Reaching agreements through argumentation: a logical model and implementation. Artificial Intelligence, 104 (1998).
11. Malheiro, B. and Oliveira, E.; Consistency and Context Management in a Multi-Agent Belief Revision Tesbed. Wooldridge, M., Müller, J. P. and Tambe, M. eds. Agent Theories, Architectures and Languages, Lecture Notes in Artificial Intelligence, Vol. 1037. Springer-Verlag, Berlin Heidelberg New York (1996).
12. Malheiro, B., Oliveira, E.; Environmental Decision Support: a Multi-Agent Approach, The First International Conference on Autonomous Agents (Agents'97), Marina del Rey, California, USA (1997).
13. Malheiro, B., Oliveira, E.; Solving Conflicting Beliefs with a Distributed Belief Revision Approach, Monard, M. C., Sichman, J. S. eds. Proceedings of The International Joint Conference 7th Ibero-American Conference on AI 15th Brazilian Symposium on AI (IBERAMIA-SBIA'2000), Atibaia, S. Paulo, Brazil. Lecture Notes in Artificial Intelligence, Vol. 1952. Springer-Verlag, Berlin Heidelberg New York (2000).

14. Oliveira, E., Mouta, F. and Rocha, A. P.; Negotiation and Conflict Resolution within a Community of Cooperative Agents. In Proceedings of the International Symposium on autonomous Decentralized Systems, Kawasaki, Japan (1993).
15. Parsons, S., Sierra, C. and Jennings, N. R. 1998. Agents that reason and negotiate by arguing, Journal of Logic and computation 8(3)(1998), 261-292.
16. Sycara, K. 1990. Persuasive Argumentation in negotiation, Theory and Decision. 28:203-242.
17. Vreeswijk, A. W. 1997. Abstract argumentation systems. Artificial Intelligence 90(1-2).
18. Wittig, T. ed. 1992. ARCHON: An Architecture for Multi-Agent Systems, Ellis Horwood.

Scheduling, Re-scheduling and Communication in the Multi-agent Extended Enterprise Environment

Joaquim Reis[1], Nuno Mamede[2]

[1] ISCTE, Dept. Ciências e Tecnologias de Informação, Avenida das Forças Armadas, 1649-026 Lisboa, Portugal
Joaquim.Reis@iscte.pt
[2] IST, Dept. de Engenharia Informática, Avenida Rovisco Pais, 1049-001 Lisboa, Portugal
Nuno.Mamede@acm.org

Abstract. In this article we describe a multi-agent dynamic scheduling environment where autonomous agents represent enterprises and manage the capacity of individual macro-resources in a production-distribution context. The agents are linked by client-supplier relationships and inter-agent communication must take place. The model of the environment, the appropriate agent interaction protocol and a cooperative scheduling approach, emphasizing a temporal scheduling perspective of scheduling problems, are described. The scheduling approach is based on a coordination mechanism supported by the interchange of certain temporal information among pairs of client-supplier agents involved. This information allows the agents to locally perceive hard global temporal constraints and recognize non over-constrained problems and, in this case, rule out non temporally-feasible solutions and establish an initial solution. The same kind of information is then used to guide re-scheduling to repair the initial solution and converge to a final one.

Keywords. Scheduling, Multi-Agent Systems, Supply-Chain Management.

1 Introduction

Scheduling is the allocation of resources over time to perform a collection of tasks, subject to temporal and capacity constraints. In classical/Operations Research (OR) scheduling approaches, a centralized perspective is assumed: all problem data is known by a central entity and scheduling decisions are taken by the same entity, based on a well defined criteria. Sometimes, in more modern Artificial Intelligence (AI), or mixed OR/AI, based approaches, the same kind of centralized perspective is assumed too. For a general introduction to OR approaches to scheduling problems see [9] or [2]; AI based approaches can be found in [5], [25] or [10], for instance. Planning and coordination of logistics activities has been, in the areas of OR/Management Science, the subject of investigation since the sixties [8]. The problem of scheduling in this kind of environments has had, recently, a more dedicated attention; see [6], [1], [11] or [15], for instance. In this article, the specific logistics context of the supply-chain/Extended Enterprise (EE) [14] is considered, for the short-term activity of scheduling of production-distribution tasks. The EE is usually assumed to be a kind of cooperative Virtual Organization, or Virtual Enterprise, where the set of inter-dependent participant enterprises is relatively stable; for concepts, terminology and classification see [3]; in [4] other approaches to scheduling in this kind of context can be found. A distributed approach is more natural in this case, because scheduling data and decisions are inherently distributed, as resources are managed by individual, geographically decentralized and autonomous entities (enterprises, organizations). So, in our approach, we adopted the AI Multi-Agent Systems paradigm (see [13] or [24]). In the following, we describe ongoing investigation developing from work published on the subject of multi-agent scheduling in production-distribution environments (see [16], [17], [18], [19], [20] and [21]).

The scheduling problems we consider have the following features:

a) Communication is involved - Agents must communicate (at least to exchange product orders with clients/suppliers);
b) Cooperation is involved - Each agent must cooperate so that it won't invalidate feasible scheduling solutions;
c) Scheduling activity is highly dynamic - Problem solution development is an on-going process during which unforeseen events must always be accommodated, and re-scheduling and giving up a scheduling problem are options to be considered.

The following sections present: a brief description of the model of the multi-agent scheduling environment (sec.2), the agent interaction protocol (sec.3), the cooperative approach proposed for multi-agent scheduling problems (sec.4), the initial scheduling step (sec.5) and the re-scheduling (sec.6), both from an individual agent perspective, and finally, future work and conclusions (sec.7). Secs. 5 and 6 are presented with examples based on simulations.

2 The Scheduling Environment Model

In past work (referred above) we have proposed a model of the EE scheduling environment based on an *agent network*, with each agent managing an aggregate scheduling resource, representing a *production*, a *transportation*, or a *store* resource, and linked through client-supplier relationships. A scheduling resource is just an individual node of a *physical network* of resources, and accommodates the representation of the agent tasks scheduled and the available capacity along a certain scheduling temporal horizon. Ordinary network agents are named *capacity*, or *manager*, *agents*, because they are responsible for managing the capacity of a resource, and they can be *production*, *transportation* or *store* class agents; production and transportation class agents are grouped under the *processor* agent class, because the capacity they manage is based on a rate. A special *supervision agent*, with no resource, plays the role of an interface with the outside, and can introduce new scheduling problems into the agent network.

Pairs of client-supplier capacity agents can communicate, basically through the exchange of product request messages (see next section), which contain product, quantity of product and proposed due-date information. The supervision agent communicates with special agents playing the roles of *retail* and *raw-material* agents, located at the downstream and the upstream end of the agent network (which are pure clients and pure suppliers for the network), respectively.

A scheduling problem is defined by a *global product request from the outside* of the agent network (*i.e.*, a request of a final product of the network), the global due-date `DD`, and the global release date `RD`. These two dates are the limits of the scheduling temporal horizon and are considered the hard global temporal constraints of the problem. The supervision agent introduces a new scheduling problem by communicating a global product request from outside to the appropriate retail agent (networks can be multi-product so, the set of products deliverable can depend on the retail agent); later, after the capacity agents have propagated among them, upstream the agent network, the necessary *local product requests*, the supervision agent will collect *global product requests to outside* from raw-material agents. A scheduling problem will cease to exist in the network when the time of the last of the local due-date comes, or if some local requests are rejected, or accepted and then canceled (the supervision agent knows this as messages of satisfaction, rejection and cancellation will be propagated to retail and raw-material agents and then communicated to it).

In order to satisfy a product request from a client a capacity agent must schedule a task on its resource. A task needs a supply of one (in the case of a store or transportation agent), or one or more products (in the case of a production agent and depending on the components/materials for the task product). For those supplies the agent must send the appropriate product requests to the appropriate suppliers.[1] The task consumes a non-changing amount of resource capacity during its temporal interval. The duration of the task depends on the product, the quantity of product and additional parameters related to the characteristics of the resource and, in the case of processor agents, to the amount of capacity dedicated to the task.[2] The details of these latter parameters are omitted here to allow simplicity of explanation, and we will assume a non-changing duration for all tasks, except for store tasks (with flexible duration, and minimum of 1 time unit).

Although tasks are private to the agents (only the communication messages are perceived by more than one agent), we can view the set of tasks that some agents of the network schedule to satisfy a global product request from outside, as whole, as belonging to a *network job*, see example in Fig. 1. This is a just an analogue of the concept of job used in classical production scheduling problems.

[1] We assume that there is, for each capacity agent, a unique supplier for each supply product. As a result of this simplifying assumption, the lack of a product supply has, as a final result, the network being unable to satisfy a global product request from the outside. Allowing multiple suppliers for the same supply product opens the door to another issues (like choosing the preferred supplier, possibly with negotiation based on prices, or due-dates), in which we are not interested, for now.

[2] Basically, more capacity invested gives a shorter task duration.

A *solution* for a scheduling problem is a set of product requests agreed by pairs of client-supplier agents and the set of agent tasks, necessary to satisfy the global product request given by the problem, forming the corresponding network job. A *feasible solution* has nor temporal nor capacity conflicts, *i.e.*, it respects both all temporal and all capacity constraints. For capacity constraints to be respected, no

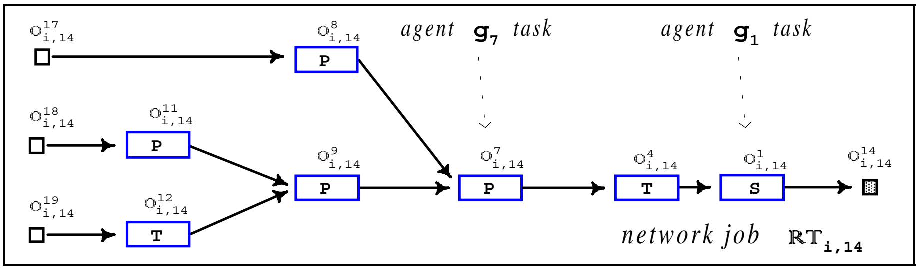


Fig. 1. A network job (job $RT_{i,14}$). The task of an agent g_k for the i^{th} global request to retail agent g_r is denoted by $O^{k}_{i,r}$ (and this task belongs to the network job denoted by $RT_{i,r}$); P, T and S denote production, transportation and store tasks, respectively.

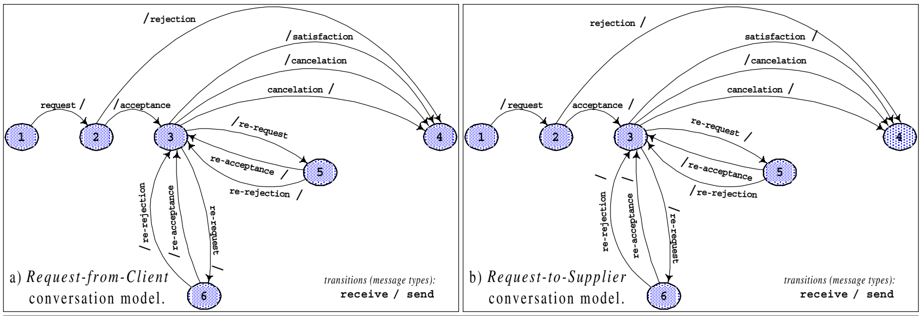
capacity over-allocation must occur with any task of the solution, for any agent, at any moment of the scheduling temporal horizon. For temporal constraints to be respected, all local product requests of the solution must fall within the global release date and global due-date of the problem; also, for each agent, the interval of its task must fall in between the due-date agreed for the client request and (the latest of) the due-date(s) agreed for the request(s) made to the supplier(s), and the latter due-date(s) must precede the former.

More details on the resources and the physical network are given in [16] and [18]; about the agent network and agent architecture see [17], [18], and [19].

3 The Agent Interaction Protocol

In this section we expose the high level inter-agent protocol used for scheduling in the EE network.

The agent interaction activity for scheduling occurs through the interchange of messages, of predetermined types, between pairs of client-supplier agents. The exchange of a message always occurs in the context of a *conversation* between the sender and the receiver agents. A conversation has a *conversation model*, which contains information about the predetermined sequences of the types of messages exchangeable and is defined through a finite state machine. An *interaction protocol* is defined as a set of conversation models. For the interaction of capacity agents we defined the *Manager-Manager* interaction protocol, represented in Fig. 2.



`request` - product request, sent by a client agent to a supplier agent.	`rejection` - rejection of a previously received product request, sent by the supplier to the client.	`re-request` - re-scheduling request, sent by the supplier (client) to the client (supplier), asking to re-schedule a previously accepted product request to a given due-date.	`re-rejection` - rejection of a previously received re-scheduling request, sent by the receiver to the sender of the re-scheduling request.	`cancelation` - signals giving up a previously accepted product request, sent by the supplier (client) to the client (supplier).
c) Message types and description.	`acceptance` - acceptance of a previously received product request, sent by the supplier to the client.		`re-acceptance` - acceptance of a previously received re-scheduling request, sent by the receiver to the sender of the re-scheduling request.	`satisfaction` - signals delivery of a previously accepted product request, sent by supplier to client at the time of the due-date of the product request.

Fig. 2. Conversation models and messages for the *Manager-Manager* interaction protocol.

The protocol has the associated conversation models *Request-from-Client* and *Request-to-Supplier*, described as finite state machine diagrams in Fig. 2-a and Fig. 2-b, and to be used by an agent when playing the roles of a supplier and a client agent, respectively. Fig. 2-c describes the types of messages exchangeable.

4 A Cooperative Multi-agent Scheduling Approach

Classically, scheduling is considered a difficult problem [7]. In general, solutions for a scheduling problem have to be searched for, and the search space can be very large. Additionally, for a multi-agent scheduling problem, a part of the effort is invested in coordination of the agents involved which, in our case, means sharing information through message exchange. Message exchange is considered costly so, methods of pruning the search space for finding at least a first feasible solution with minimal coordination efforts are considered satisfactory.

The approach we propose in this article, for the cooperative individual (agent) scheduling behavior, is a *minimal approach*, that is, an agent viewing a scheduling problem solution as feasible won't do anything respecting to the scheduling problem. A first version of this approach appeared in [20]; in [21] we presented a refined approach only for processor, *i.e.*, production or transportation, agents; in the present article we cover also store agents and, additionally, include the respective *minimal* re-scheduling actions for processor and store agent cases.

Consider two sets of solutions for a scheduling problem: the set of *time-feasible solutions*, which respect all temporal constraints, and the set of *resource-feasible solutions*, which respect all capacity constraints. A feasible solution is one that is both time-feasible and resource-feasible so, the set of feasible solutions is the intersection of those two sets. A problem is *temporally over-constrained* if the set of time-feasible solutions is empty, and is *resource over-constrained* if the set of resource-feasible solutions is empty. If a problem has both non-empty time-feasible and resource-feasible solution sets, and their intersection is non-empty, then the problem has feasible solutions. We propose an approach using the following three step procedure, for each individual capacity agent:

Step 1. Acceptance and initial solution - Detect if the problem is temporally over-constrained, and if it isn't, establish an initial solution, and proceed in the next step; if it is, terminate the procedure by rejecting the problem, because it has no feasible solution;

Step 2. Re-schedule to find a time-feasible solution - If the established solution is time-feasible, proceed in the next step; if it isn't, re-schedule to remove all temporal conflicts;

Step 3. Re-schedule to find a feasible solution - For a resource-feasible solution, terminate the procedure; otherwise, try to re-schedule to remove all capacity conflicts without creating temporal conflicts, resorting to cancellation, with task un-scheduling, as a last choice, if necessary.

As some approaches in the literature, this procedure starts by establishing one initial, possibly non-feasible, solution which is then repaired in order to remove conflicts; see, for instance, [12]. Steps 1 and 2 of the procedure are oriented for a temporal scheduling perspective and concern only to a single scheduling problem of the agent. Step 3 is oriented for a resource scheduling perspective and can involve all scheduling problems of the agent at step 3, as all tasks of the agent compete for the agent resource capacity.

5 Step 1: Scheduling an Initial Solution

We now show, through examples for processor and store class agents, how an agent can locally recognize a non temporally over-constrained problem and, in that case, contribute to establish an initial solution (step 1).

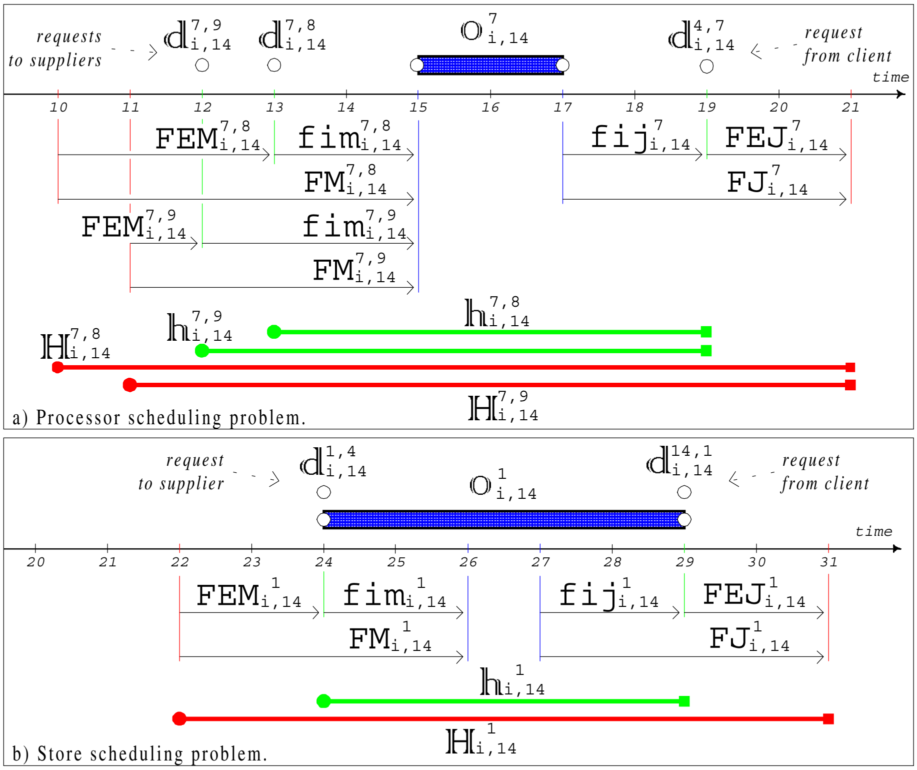
For a processor agent, suppose an agent $\mathtt{g}_7$ has a scheduling problem with the processor task $\mathbb{O}^{7}_{\mathtt{i,14}}$ of network job in Fig. 1. In Fig. 3-a, a possible situation for this task (which also represents a feasible solution for the problem) is represented on a timeline. As $\mathtt{g}_7$ has two suppliers for the task, two requests to two suppliers are shown, $\mathbb{d}^{7,8}_{\mathtt{i,14}}$ and $\mathbb{d}^{7,9}_{\mathtt{i,14}}$, besides the request from the client, $\mathbb{d}^{4,7}_{\mathtt{i,14}}$. Intervals (denoted by $\mathbb{h}$ and $\mathbb{H}$ symbols) and temporal slacks are shown in Fig. 3-a. Symbols `fij`, `fim`, `FEJ`, `FEM`, `FJ` and `FM` denote, respectively, the *internal downstream slack*, *internal upstream slacks*, *external downstream slack*, *external upstream slacks*, *downstream slack* and *upstream slacks*. For each kind of upstream slacks there is one per each supplier (slacks are represented by arrows). By definition:

$$\mathtt{FJ}^{7}_{i,14}=\mathtt{FEJ}^{7}_{i,14}+\mathtt{fij}^{7}_{i,14},\quad \mathtt{fij}^{7}_{i,14}=\mathtt{TIME}(\mathbb{d}^{4,7}_{i,14})-\mathtt{ENDTIME}(\mathbb{O}^{7}_{i,14})$$

$$\mathtt{FM}^{7,j}_{i,14}=\mathtt{FEM}^{7,j}_{i,14}+\mathtt{fim}^{7,j}_{i,14},\quad \mathtt{fim}^{7,j}_{i,14}=\mathtt{STARTTIME}(\mathbb{O}^{7}_{i,14})-\mathtt{TIME}(\mathbb{d}^{7,j}_{i,14})$$

$$(\mathtt{j=8,9})$$

Internal slacks are inserted locally, by the initiative of the agent, when scheduling the task and making requests to suppliers; external slacks are imposed by the other agents of the network. It is assumed that, in any case, *the agent will maintain non negative internal slacks*. Each of the $\mathbb{h}$'s is an interval between one of the supplier due-dates and the client due-date (13 and 19, and 12 and 19, in Fig. 3-a); each of the $\mathbb{H}$'s is an interval between one of the earliest start times and the latest finish time for the task (10 and 21, and 11 and 21, in Fig. 3-a). Each of the latter pairs of temporal



requests to suppliers
$\mathbb{d}^{7,9}_{i,14}$
$\mathbb{d}^{7,8}_{i,14}$
$\mathbb{O}^{7}_{i,14}$
$\mathbb{d}^{4,7}_{i,14}$
request from client
time
10 11 12 13 14 15 16 17 18 19 20 21
$\mathtt{FEM}^{7,8}_{i,14}$
$\mathtt{fim}^{7,8}_{i,14}$
$\mathtt{FM}^{7,8}_{i,14}$
$\mathtt{FEM}^{7,9}_{i,14}$
$\mathtt{fim}^{7,9}_{i,14}$
$\mathtt{FM}^{7,9}_{i,14}$
$\mathtt{fij}^{7}_{i,14}$
$\mathtt{FEJ}^{7}_{i,14}$
$\mathtt{FJ}^{7}_{i,14}$
$\mathbb{h}^{7,8}_{i,14}$
$\mathbb{h}^{7,9}_{i,14}$
$\mathbb{H}^{7,8}_{i,14}$
$\mathbb{H}^{7,9}_{i,14}$
a) Processor scheduling problem.
request to supplier
$\mathbb{d}^{1,4}_{i,14}$
$\mathbb{O}^{1}_{i,14}$
$\mathbb{d}^{14,1}_{i,14}$
request from client
time
20 21 22 23 24 25 26 27 28 29 30 31
$\mathtt{FEM}^{1}_{i,14}$
$\mathtt{fim}^{1}_{i,14}$
$\mathtt{FM}^{1}_{i,14}$
$\mathtt{fij}^{1}_{i,14}$
$\mathtt{FEJ}^{1}_{i,14}$
$\mathtt{FJ}^{1}_{i,14}$
$\mathbb{h}^{1}_{i,14}$
$\mathbb{H}^{1}_{i,14}$
b) Store scheduling problem.

Fig. 3. Scheduling problem parameters: a) for processor agent $\mathtt{g}_7$, and b) for store agent $\mathtt{g}_1$ (for the values in the timelines of these two cases no relationship is intended).

points are hard temporal constraints for one of the former pairs of due-dates. Also, the temporal end points of the most restrictive $\mathbb{H}$ interval (11 and 21 in Fig. 3-a) are hard temporal constraints for the task; let us denote these by $\mathtt{RD}^{7}_{i,14}$ and $\mathtt{DD}^{7}_{i,14}$, for the upstream and downstream point, respectively.

It is easy to see that, in order for a solution to be time-feasible, the interval of the task must be contained in the most restricted $\mathbb{h}$ interval, and each $\mathbb{h}$ interval must be contained in the corresponding (same supplier) $\mathbb{H}$ interval. For this to hold, no temporal slack can be negative. Also, if the duration of the most restrictive $\mathbb{H}$ interval is less than the task duration, the problem is temporally over-constrained, and the agent can reject it.

We propose that product request messages from the client, additionally to product request information, *carry the value of the* FEJ *slack*; also, request acceptance messages from suppliers will *carry the value of the respective* FEM *slack*. In our example, agent g_7 would then calculate the $\mathtt{DD}^{7}_{i,14}$ and $\mathtt{RD}^{7}_{i,14}$ values by:

$$\mathtt{DD}^{7}_{i,14}=\mathtt{TIME}(\mathbb{d}^{4,7}_{i,14})+\mathtt{FEJ}^{7}_{i,14} \quad \text{and} \quad \mathtt{RD}^{7}_{i,14}=\underset{j=8,9}{\mathtt{MAX}}(\mathtt{RD}^{7,j}_{i,14})$$

where: $\mathtt{RD}^{7,j}_{i,14}=\mathtt{TIME}(\mathbb{d}^{7,j}_{i,14})-\mathtt{FEM}^{7,j}_{i,14} \quad (j=8,9)$

When the agent receives request $\mathbb{d}^{4,7}_{i,14}$ from the client, it will, guaranteeing non-negative values of fij and fim for the task to be scheduled, make requests $\mathbb{d}^{7,8}_{i,14}$ and $\mathbb{d}^{7,9}_{i,14}$ to suppliers, passing them also the (supplier FEJ) value $\mathtt{FJ}^{7}_{i,14}+\mathtt{fim}^{7,j}_{i,14}$ (for $j=8,9$). When the agent receives all the request acceptances from the suppliers, verifies first if the problem is temporally over-constrained, by testing if $\mathtt{DD}^{7}_{i,14}-\mathtt{RD}^{7}_{i,14} < \mathtt{DURATION}(\mathbb{O}^{7}_{i,14})$. If this is true the problem must be rejected. Otherwise, the agent will send the acceptance message to the client, passing it also the (client FEM) value $\mathtt{FM}^{7}_{i,14}+\mathtt{fij}^{7}_{i,14}$, where $\mathtt{FM}^{7}_{i,14}=\mathtt{STARTIME}(\mathbb{O}^{7}_{i,14})-\mathtt{RD}^{7}_{i,14}$. If step 1 concludes with a non temporally over-constrained problem, agent g_7 will internally keep the tuple $<\mathtt{DD}^{7}_{i,14},\{\mathtt{RD}^{7,8}_{i,14},\mathtt{RD}^{7,9}_{i,14}\},\mathbb{d}^{4,7}_{i,14},\{\mathbb{d}^{7,8}_{i,14},\mathbb{d}^{7,9}_{i,14}\},\mathbb{O}^{7}_{i,14}>$, which represents the agent local perspective of the scheduling problem, and also includes (a part of) the initial solution.

For a store agent, suppose an agent g_1 has a scheduling problem with the store task $\mathbb{O}^{1}_{i,14}$ of network job in Fig. 1. In Fig. 3-b, a possible situation for this task (which also represents a feasible solution for the problem) is represented on a timeline. The case is similar to the one for agent g_7, with the exceptions described in the following. There is only one request to a supplier, as g_1 is a store agent. The internal slacks are defined differently: the task interval is equal to the (unique) $\mathbb{h}$ interval, and part of the task duration is considered as internal slack (if its duration is greater than 1, which is the minimum assuming the task must exist), with the relationship $\mathtt{fij}^{1}_{i,14}+\mathtt{fim}^{1}_{i,14}=\mathtt{DURATION}(\mathbb{O}^{1}_{i,14})-1$ always holding. Fig. 3-b suggests *symmetrical* definitions for fij and fim slacks, with the minimum duration interval "centered" in the $\mathbb{h}$ interval but, for purposes of temporal constraint violation identification (see next section), the following definitions must be used. For the downstream side violations (cases 1 and 3, in the next section), `fij` and `fim` are defined by (the minimum duration interval is shifted to the left):

$\mathtt{fij}^{1}_{i,14}=\mathtt{DURATION}(\mathbb{O}^{1}_{i,14})-1$, and $\mathtt{fim}^{1}_{i,14}=0$

For the upstream side violations (cases 2 and 4, in the next section), `fij` and `fim` are defined by (the minimum duration interval is shifted to the right):

$\mathtt{fij}^{1}_{i,14}=0$, and $\mathtt{fim}^{1}_{i,14}=\mathtt{DURATION}(\mathbb{O}^{1}_{i,14})-1$

The problem is temporally over-constrained if $\mathtt{DD}^{7}_{i,14}-\mathtt{RD}^{7}_{i,14} < 1$. The values of $\mathtt{FEJ}^{1}_{i,14}+\mathtt{DURATION}(\mathbb{O}^{1}_{i,14})-1$ and $\mathtt{FEM}^{1}_{i,14}+\mathtt{DURATION}(\mathbb{O}^{1}_{i,14})-1$ must be passed to the supplier and to the client (as the supplier `FEJ` value, and the client `FEM` value), respectively. Finally, if step 1 concludes with a non temporally over-constrained problem, agent g_1 will keep the tuple $<\mathtt{DD}^{1}_{i,14}, \{\mathtt{RD}^{1}_{i,14}\}, \mathbb{d}^{14,1}_{i,14}, \{\mathbb{d}^{1,4}_{i,14}\}, \mathbb{O}^{1}_{i,14}>$.

6 Step 2: Re-scheduling for a Time-Feasible Solution

In this section we show how, starting from an initial solution with temporal conflicts, agents g_7 and g_1 can locally contribute to obtain a time-feasible solution (step 2).

For the local situations represented in Fig. 3-a and Fig. 3-b, all slacks are positive so, the solution is seen as temporally-feasible (by agent g_7, and agent g_1, respectively). In these cases, an agent will do nothing, unless it receives any re-scheduling request, which it could accept provided non-negative internal slacks can be maintained, for a processor agent, or a task with duration greater than 0 is possible, in the case of a store agent. Otherwise, four kinds of possible local situations can occur where the agent itself must take the initiative of some re-scheduling actions.

The situations referred are described by the re-scheduling cases 1, 2, 3 and 4, for which we show examples in Fig. 4 for processor agents (for agent g_7), and in Fig. 5 for store agents (for agent g_1). Each figure represents, for each case: a) the situation detected, and b) the situation after a minimal re-scheduling action. No relationship is intended among timeline values of the processor and the store agent cases. In the text following, upper indexes are omitted in slack symbols in order to cover both, processor and store, agent cases. Cases 1 and 2 *must be considered first*, by the agent.

Case 1 occurs if $\mathtt{FJ}_{i,14}<0$ *and* $\mathtt{FEJ}_{i,14}<0$ (the task and the client request violate the hard temporal constraint downstream, 17 in Fig. 4-1-a, and 20 in Fig. 5-1-a). The detection of case 1 must be followed by the appropriate task re-scheduling and client request re-scheduling to earlier times (resulting in the situation shown in Fig. 4-1-b, and Fig. 5-1-b). Re-scheduling of some requests to suppliers can (or cannot) then be necessary to maintain non-negative `fim` slacks, at the upstream side.

Case 2 occurs if, for some supplier, $\mathtt{FM}_{i,14} < 0$ *and* $\mathtt{FEM}_{i,14} < 0$, (the task and

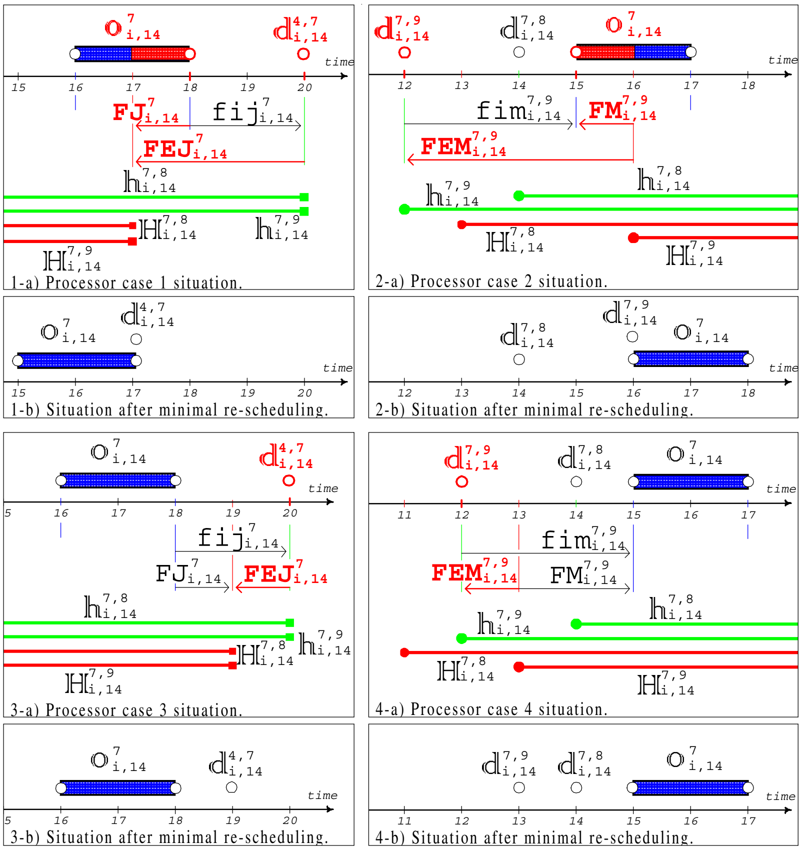


Fig. 4. Examples of re-scheduling cases 1, 2, 3 and 4, for a processor agent, with situations before, and after, minimal re-scheduling.

some requests to suppliers violate hard temporal constraints upstream, 16 in Fig. 4-2-a, and 29 in Fig. 5-2-a). The detection of case 2 must be followed by the appropriate task re-scheduling and the re-scheduling of the offending requests to suppliers to later times (resulting in the situation shown in Fig. 4-2-b, and Fig. 5-2-b). Re-scheduling of the client request can (or cannot) then be necessary to maintain a non-negative `fij` slack, at the downstream side.

After handling cases 1 and 2, cases 3 and 4 are handled.

Case 3 occurs if $\mathtt{FJ}_{i,14} < 0$ (the client request violates the hard temporal constraint downstream, 19 in Fig. 4-3-a, and 26 in Fig. 5-3-a). The detection of case

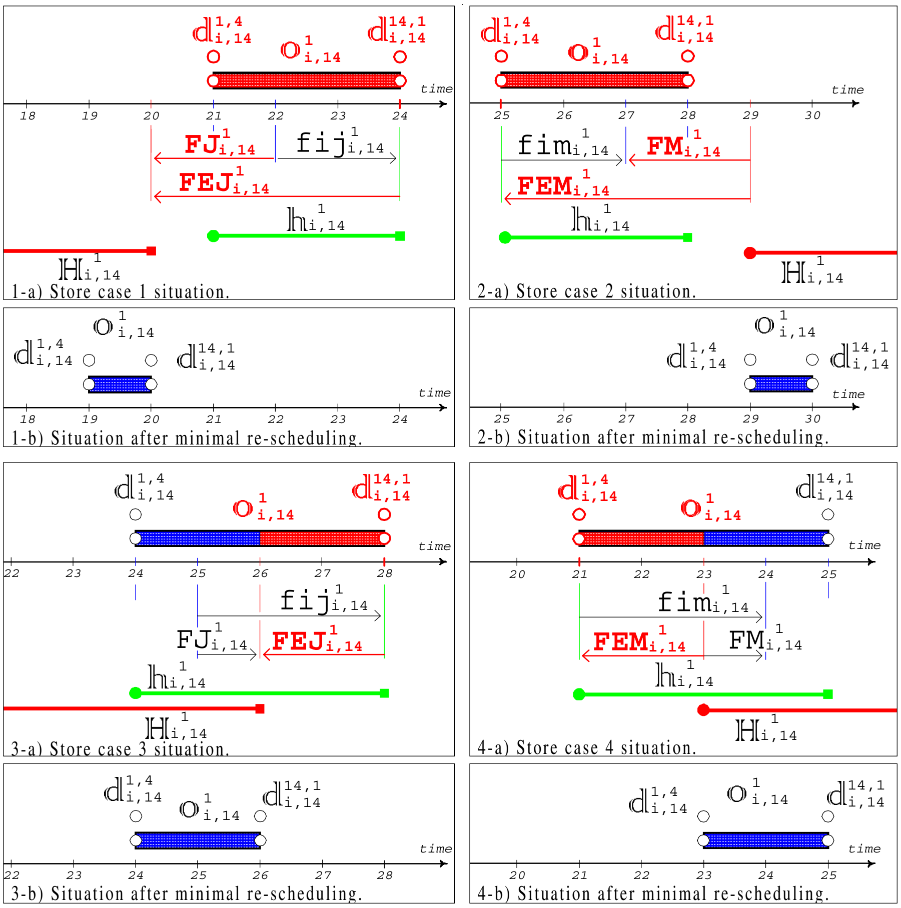

1-a) Store case 1 situation.
2-a) Store case 2 situation.
1-b) Situation after minimal re-scheduling.
2-b) Situation after minimal re-scheduling.
3-a) Store case 3 situation.
4-a) Store case 4 situation.
3-b) Situation after minimal re-scheduling.
4-b) Situation after minimal re-scheduling.
time

Fig. 5. Examples of re-scheduling cases 1, 2, 3 and 4, for a store agent, with situations before, and after, minimal re-scheduling.

3 must be followed by the appropriate client request re-scheduling to an earlier time (resulting in the situation shown in Fig. 4-3-b, and Fig. 5-3-b).

Case 4 occurs if, for some supplier, $\mathtt{FEM}_{i,14} < 0$ (some requests to suppliers violate hard temporal constraints upstream, 13 in Fig. 4-4-a, and 23 in Fig. 5-4-a). The detection of case 4 must be followed by the appropriate re-scheduling of the offending requests to suppliers to later times (resulting in the situation shown in Fig. 4-4-b, and Fig. 5-4-b).

7 Conclusion and Future Work

We described a multi-agent dynamic scheduling environment involving communication and cooperation, and an approach for multi-agent cooperative scheduling based on a three step procedure for individual agents. Step 1 allows agents to detect locally if the problem is temporally over-constrained and, in the case it isn't, schedule an initial, possibly non time-feasible, solution. By locally exchanging specific temporal slack values, agents are able to locally perceive the hard global temporal constraints of a problem, and rule out non time-feasible solutions in the subsequent steps. Each of the slack values exchanged in step 1 corresponds, for a particular agent, to a sum of slacks, downstream and upstream the agent network, and so, they cannot be considered private information of any agent in particular. In step 2, if necessary, agents repair the initial solution to obtain a time-feasible one.

The procedure is very general with respect to its step 3. This step can be refined to accommodate additional improved coordination mechanisms for implementing certain search strategies, based on capacity/resource constrainedness (*e.g.*, see [23] or [22]), leading the agents on a fast convergence to specific feasible solutions. For instance, feasible solutions satisfying some scheduling preferences or optimizing some criteria, either from an individual agent perspective, or from a global one, or both. This is a subject for our future work.

References

1. Arnold, Jörg, et al, *Production Planning and Control within Supply Chains*, ESPRIT project 20544, X-CITTIC, 1997.
2. Blazewicz, J.; Ecker, K.H.; Schmidt, G.; Weglarz, J., *Scheduling in Computer and Manufacturing Systems*, Springer Verlag, 1994
3. Camarinha-Matos, Luís M.; Afsarmanesh, Hamideh, *The Virtual Enterprise Concept*, in Infrastructures for Virtual Enterprises, Camarinha-Matos, L.M. and Afsarmanesh, H. (eds.), Kluwer Academic Publishers, Dordrecht, The Netherlands, 1999, 3-14.
4. Camarinha-Matos, Luís M.; Afsarmanesh, Hamideh, *Infrastructures for Virtual Enterprises, Networking Industrial Enterprises*, Luís M. Camarinha-Matos, Hamideh Afsarmanesh (eds.), Kluwer Academic Publishers, Dordrecht, The Netherlands, 1999.
5. Dorn, J.; Froeschel, K., *Scheduling of Production Processes*, Dorn, J.; Froeschel, K. (eds.), Elis Horwood, Ld., 1993.
6. Fox, Mark S., et al, *The Integrated Supply Chain Management System*, Department of Industrial Engineering, University of Toronto, Canada, 1993.
7. Garey, M.R.; Johnson, D.S., *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W.H. Freeman and Co., New York, 1979.
8. Graves, S.C.; Kan, A.H.G. Rinnooy; Zipkin, P.H., (eds.), *Logistics of Production and Inventory*, Handbooks in Operations Research and Management Science, Volume 4, North-Holland, Amsterdam, 1993.
9. Kan, A.H.G. Rinnooy, *Machine Scheduling Problems*, Martinus Nijhoff, The Hague, 1976.

10. Kerr, Roger; Szelke, Elizabeth (eds.), *Artificial Intelligence in Reactive Scheduling*, Chapman & Hall, 1995.
11. Kjenstad, Dag, *Coordinated Supply Chain Scheduling*, PhD. Thesis, Norwegian University of Science and Technology, Trondheim, Norway, 1998.
12. Minton, Steven, et al, *Minimizing Conflicts: a Heuristic Repair Method for Constraint Satisfaction and Scheduling Problems*, Artificial Intelligence 58, 1992, 161-205.
13. O'Hare, G.M.P.; Jennings, N.R., *Foundations of Distributed Artificial Intelligence*, John Wiley & Sons, Inc., 1996, New York, USA.
14. O'Neill, H.; Sackett, P., *The Extended Enterprise Reference Framework*, Balanced Automation Systems II, Camarinha-Matos, L.M. and Afsarmanesh, H. (eds.), 1996, Chapman & Hall, London, UK, 401-412
15. Rabelo, Ricardo J.; Camarinha-Matos, Luis M.; Afsarmanesh, Hamideh, *Multiagent Perspectives to Agile Scheduling*, Basys'98 International Conference on Balanced Automation Systems, Prague, Czech Republic, 1998.
16. Reis, J.; Mamede, N.; O'Neill, H., *Ontologia para um Modelo de Planeamento e Controlo na Empresa Estendida*, Proceedings of the IBERAMIA'98, Lisbon, Portugal, 1998, Helder Coelho (ed.), Edições Colibri, Lisbon, Portugal, 43-54 (in portuguese).
17. Reis, J.; Mamede, N.; O’Neill, H., *Agent Communication for Scheduling in the Extended Enterprise*, Proceedings of the IFIP TC5 WG5.3/PRODNET Conference on Infrastructures for Virtual Enterprises, Porto, Portugal, 1999, Camarinha-Matos, L.M., Afsarmanesh H. (eds.), Kluwer Academic Publishers, Dordrecht, The Netherlands, 1999, 353-364.
18. Reis, J.; Mamede, N., *What’s in a Node, Nodes and Agents in Logistic Networks*, Proceedings of the ICEIS’99, 1st International Conference on Enterprise Information Systems, Setúbal, Portugal, 1999, Filipe, J. and Cordeiro, J. (eds.), 285-291.
19. Reis, J.; Mamede, N., *An Agent Architecture for Multi Agent Dynamic Scheduling*, Proceedings of the ICEIS’2000, 2nd. International Conference on Enterprise Information Systems, Stafford, UK, 2000, Sharp, B., Cordeiro, J. and Filipe, J. (eds.), 203-208.
20. Reis, J.; Mamede, N.; O’Neill, H., *Locally Perceiving Hard Global Constraints in Multi-Agent Scheduling*, Journal of Intelligent Manufacturing, Vol. 12, No. 2, 2001, 227-240.
21. Reis, J.; Mamede, N., *Multi-Agent Dynamic Scheduling and Re-Scheduling with Global Temporal Constraints*, Proceedings of the ICEIS’2001, 3rd International Conference on Enterprise Information Systems, Setúbal, Portugal, 2001, Miranda, P., Sharp, B., Pakstas, A. and Filipe, J (eds.), Vol. I, 315 321.
22. Sadeh, Norman, *Micro-Oportunistic Scheduling: The Micro-Boss Factory Scheduler*, Intelligent Scheduling, Morgan Kaufman, 1994, Chapter 4.
23. Sycara, Katia P.; Roth, Steven F.; Sadeh, Norman; Fox, Mark S., *Resource Allocation in Distributed Factory Scheduling*, IEEE Expert, February, 1991, 29-40.
24. Weiss, Gerhard (ed.), *Multiagent Systems, A Modern Approach to Distributed Artificial Intelligence*, The MIT Press, 1999.
25. Zweben, Monte; Fox, Mark S., *Intelligent Scheduling*, Morgan Kaufmann Publishers, Inc., San Francisco, California, 1994.

Electronic Institutions as a Framework for Agents' Negotiation and Mutual Commitment

Ana Paula Rocha, Eugénio Oliveira

LIACC, Faculty of Engineering, University of Porto,
R. Dr. Roberto Frias, 4200-465 Porto, Portugal
{arocha,eco}@fe.up.pt

Abstract. Electronic transactions are of increasing use due to its openness and continuous availability. The rapid growth of information and communication technologies has helped the expansion of these electronic transactions, however, issues related to security and trust are yet limiting its action space, mainly in what concerns business to business activity. This paper introduces an Electronic Institution framework to help in electronic transactions management making available norms and rules as well as monitoring business participants' behaviour in specific electronic business transactions. Virtual Organisation (VO) life cycle has been used as a complex scenario encompassing electronic transactions and where Electronic Institutions help in both formation and operation phase. A flexible negotiation process that includes multi-attribute and learning capabilities as well as distributed dependencies resolution is here proposed for VO formation. "Phased commitment" is another concept here introduced for VO operation monitoring through the Electronic Institution.

1 Introduction

An Electronic Institution is a framework for enabling, through a communication network, automatic transactions between parties according to sets of explicit institutional norms and rules.

An Electronic Institution helps on both providing tools and services for and on supervising the intended relationships between the parties.

Usually, parties engaged in electronic transactions and joint actions are software agents mediated. We therefore believe that an appropriate Electronic Institution can be implemented as an agent-based framework where external agents can meet together according to a set of established and fully agreed mutual constraints.

A good example illustrating the need of an Electronic Institution can be found in a software framework providing the automatic services needed for helping on the Virtual Organisations' life cycle.

A Virtual Organisation (VO) is an aggregation of autonomous and independent organisations connected through a network (possibly a public network like Internet) and brought together to deliver a product or service in response to a customer need. Virtual Organisation management should be supported by efficient information and communication technology through its entire life cycle.

Tools for Virtual Organisations formation process, through the use of an Electronic Market providing enhanced protocols for appropriate negotiation can easily be accommodate into the Electronic Institution available services. In our approach, these services include automatic capabilities for adaptive bid formulation, accepting a qualitative feedback for the sake of keeping the information as much as possible private to each one of the negotiating agents, as well as multi-issue bid evaluation.

Contrary to other approaches [1, 2] that use a-priori fixed values for weighting attributes' values in the bid evaluation function, we here advocate weighting those values, reflecting the deviation from the preferred values, according to the relative importance of the attributes. Further more, the negotiation protocol we are introducing, here called Q-Negotiation, includes the capability for Agents that are trying to supply mutually constrained items, to deal with the respective inter-dependencies while keeping their self-interestedness.

During the self-interested agents Q-Negotiation process, they may have to agree on supplying sets of mutually constrained items whose values, although not being optimum for each one of the individual agents, correspond to a minimal joint decrement of the agents' maximum utility. However, it would not be fair that this agreement, in the name of the best possible joint utility, benefits one agent more than the other. We therefore propose that, in the case of agreements based on the agents' joint utility, the agents concerned should equally distribute the joint decrement of their maximum utility. Through the calculation of appropriate compensations, agents that have the most beneficial, know they have to transfer some utility to those who have the least.

The outcome of the Virtual Organisation formation stage is, for our proposed Electronic Institution, a "phased commitment" through which different parties commit themselves to specific future actions. We intend to link commitments, which are the result of previous negotiation during the Virtual Organisation formation stage, to the next stage (VO operation) through a monitoring process.

We also introduce the concepts of both Meta Institution and Electronic Institution which is the responsible for making available functionalities including an adaptive multi-criteria based negotiation protocol together with features for solving agents mutual dependencies.

This paper includes, besides the introduction, three more sections. The next section introduces the concepts of Electronic Institutions and Meta Institutions in the context of Virtual Organisations. A third section details our proposed Q-Negotiation algorithm used in the VO formation stage. Section 4 describes phased commitments in the VO operation stage and, finally, section 5 gives some conclusions and directions for future work.

2 Electronic Institutions and Meta Institutions

2.1 Meta-Institutions

It seems intuitive that when automated agents' interactions become more sophisticated and agents' autonomy more evident, a problem related with confidence, trust and honesty may arise. Moreover, agreements like deals made by different companies or their delegates (automated or not) always claim for a common and non-ambiguous ground of understanding. We need to design a framework, accepted by all the parties, to encompass all the automatic activities that are going to take place between agents representing different individual or collective entities.

An Electronic Institution is a framework for enabling, through a communication network, automatic transactions between parties according to sets of explicit institutional norms and rules.

An Electronic Institution helps on both providing tools and services for and on supervising the intended relationships between the parties.

However, each Electronic Institution will be, at least partially, dependent on the specific application domain of business it has been designed for. Our proposal starts with the definition of a Meta-Institution, which has to be independent of the application domain.

We see a Meta-Institution as a shell for generating specific Electronic Institutions. A Meta-Institution is a set of Electronic facilities to be requested and used in order to help on the creation of suitable Electronic Institutions according to a set of established social rules and norms of behaviour. Those rules of behaviour should apply to many different kinds of (automatic) entities interacting through the web. Besides enforcing those general rules into the social interactions, a Meta-Institution also provides tools for some important stages of the interaction process.

The main goal of a Meta-Institution is, however, to be able to make available suitable Electronic Institutions that will leave all along a particular business process in a specific application domain.

Electronic transactions between distributed and autonomous entities are becoming more and more software agents mediated. We therefore believe that an appropriate Electronic Institution can be implemented as an agent-based framework where external agents can meet together according to a set of established and fully agreed norms, rules, and mutual constraints.

2.2 Virtual Organisations and Electronic Institutions

A good example illustrating the need of an Electronic Institution (EI) can be found in providing the automatic services needed for helping on the Virtual Organisations' life cycle.

This aggregation of autonomous organisations is advantageous in the sense that it will reduce complexity – today's products and services are increasingly complex and require close coordination across many different disciplines – and most important, will enable the response to rapidly changing requirements. Virtual Organisation will only exist for a temporary time duration, that is the time needed to satisfy its purpose.

The VO life cycle is decomposed in four phases [3, 4] that will also be reflected in the EI framework:

1. *Identification of Needs*: Appropriate description of the product or service to be delivered by the VO, which guides the conceptual design of the VO.
2. *Formation (Partners Selection)*: Automatic selection of the individual organisations (partners), which based in its specific knowledge, skills, resources, costs and availability, will integrate the VO.
3. *Operation*: Control and monitoring of the partners' activities, including resolution of potential conflicts, and possible VO reconfiguration due to partial failures.
4. *Dissolution*: Breaking up the VO, distribution of the obtained profits and storage of relevant information for future use of the Electronic Institution.

We expect from a Meta-Institution, the application independent framework we have defined in section 2.1, the capability to directly help on the "Identification of needs" stage of the VO life cycle, as follows:

– First helping agents in describing their needs in such a way that can be understood by other potential members that may join later a specific and appropriate Electronic Institution and
– Second, providing the searching tools to look for potential partners who know how to achieve those described needs.

The Meta-Institution will then generate that specific Electronic Institution for the particular application domain through the instantiation of some modules according to the explicit VO goals.

We can see both the general architecture and the role of a Meta-Institution in the light of the emergence of Virtual Organisations as it is depicted in the figure below:

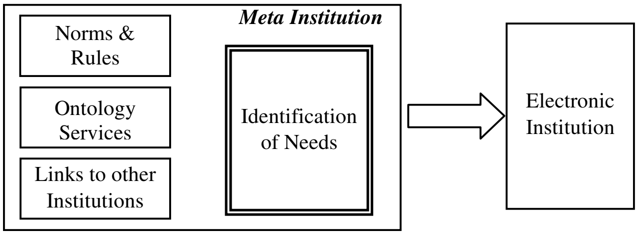


Fig. 1. General architecture and role of a Meta-Institution

The Electronic Institution, inherits appropriate rules as well as important links to other institutions that may play a crucial role for all the VO process, that are inherited from the Meta-Institution, will provide the framework for dealing with, at least, the two main stages of the VO life cycle: VO formation and VO operation monitoring.

We will describe in detail, in the next two sections, how an Electronic Institution provides the means for helping on those two stages by making available an electronic market agent-based tool.

Figure 2, below, shows the main modules of an Electronic Institution. Figure 2 also shows that VO formation module makes available our Q-Negotiation protocol enabling several organization agents (*OAgt*) to meet together through a Market (*MAgt*) leading to the VO partners selection. VO operation module uses the phased commitment mutually agreed by the partners at the end of previous stage for operation monitoring. The final module (VO dissolution) is mentioned here only for purposes of completeness.

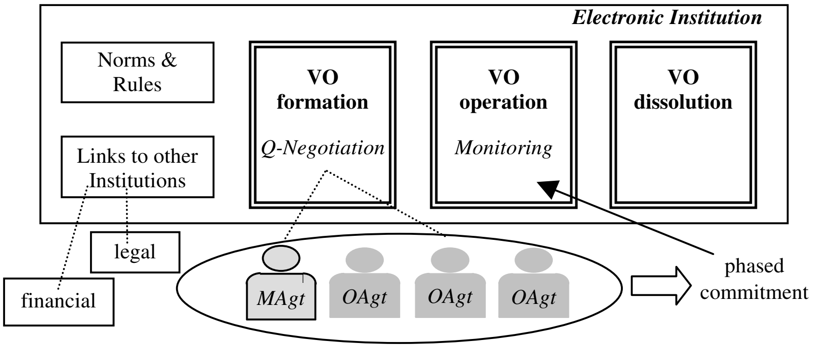


Fig. 2. General architecture and role of an Electronic Institution

3 Advanced features for Negotiation in VO formation

In the scenario of an agent-based Virtual Organisation formation process, the negotiation mechanism should enable the selection of the individual organisations that, based on their own competencies and availability, will constitute the optimal group to satisfy the previously described VO needs. In such a scenario, the adopted automatic negotiation mechanism has to be powerful enough to satisfy three important requirements:

- Capabilities for negotiating about what are the most promising organisations that should belong to VO. This means that agents, as representative of organisations, have to negotiate over goods or services those organisations are able to provide. In realistic scenarios, goods/services are described through multiple attributes, which imply that the negotiation process must be enhanced with the capability to both evaluate and formulate multi-attribute based proposals.
- Agents that are willing to belong to the Virtual Organisation may compete between them. An agent does not know other ones' negotiation strategies or even current proposals. Learning techniques can help agents to better negotiate in these partially unknown environments, by reasoning about past negotiation episodes as well as about the current one improving its own behaviour.
- In the VO formation process, each one of the individual organisations will contribute with at least one of its own capabilities (good or service) to the VO. All these contributions may be, and they usually are, mutually dependent. The negotiation process has to be able to deal with those inter-dependencies, reaching a coherent solution as the final one to be accepted by all the agents.

An important characteristic that must be considered in VO scenario is the fact that any organisation has as its main objective to maximize its own profit. In order to do that, negotiation process has to take into account agents individual rationality and self-interestedness. Keeping information private prevents loosing negotiation power to competitors, since others will never know or be able to deduce how close they are to another agent's preferences.

It is our claim that our proposed negotiation algorithm can effectively deal with these three important requirements in the VO scenario while keeping agents' information private as much as possible. In the next sections, we further detail the proposed model for agents' negotiation. First, we present a formal description of our negotiation model. Then we describe how to specifically deal with each one of the three requirements mentioned above (multi-attribute bidding, adaptive bidding and mutual dependencies resolution).

3.1 The Negotiation Model

In the VO formation process, participants in the negotiation can be either market or organisation agents. The Market Agent plays the role of organizer, meaning that it is the agent that starts and guides all the negotiation process. The Organisation Agents play the role of respondents, meaning that they are those who are willing to belong to the future VO and, therefore, they have to submit proposals during the negotiation phase.

In order to agree in a VO structure, agents (Market and several Organisations) naturally engage themselves in a sequential negotiation process composed of multiple rounds of proposals (sent by Organisations to Market) and counter-proposals which are actually comments to past proposals (sent by Market to Organisations).

The Market Agent playing a central role as organizer, models the negotiation process through the Neg^{MA} triplet as follows:

$$\mathrm{Neg}^{\mathrm{MA}} = <\mathrm{Cmpt, LAgts, H}> \qquad \textbf{(1)}$$

where:

- *Cmpt* identifies the component under negotiation
- *LAgts* is the list of respondent (organisation) agents that can provide component *Cmpt*.
- *H* is the negotiation history. Each element of H contains information related to a single negotiation round. Each negotiation round includes all proposals received during that round.

$$\mathrm{H} = \{\mathrm{H_t}\}, \qquad \mathrm{H_t} = \{<\mathrm{Prop_{ti}, Eval_{ti}}>\} \qquad \textbf{(2)}$$

$$\mathrm{Prop_{ti}} = \{\mathrm{V_1, \ldots, V_n}\}$$

where:

- $Prop_{ti}$ is the proposal sent by organisation agent i, in the negotiation round t.
- V_x is the proposal's value of attribute x.
- $Eval_{ti}$ is the evaluation value of proposal $Prop_{ti}$, from the Market Agent point of view.

Each one of the Organisation Agents model the negotiation process through the Neg^{OA} n-uple as follows:

$$\mathrm{Neg}^{OA} = <\mathrm{Cmpt, MAgt, H, Q}> \quad (3)$$

- *Cmpt* identifies the component under negotiation
- *MAgt* identifies Market Agent that is the organizer of this negotiation process.
- *H* is the negotiation history. Each element of H contains information related to a single negotiation round. Each negotiation round includes the proposal sent by each specific Organisation Agent to the Market Agent plus the feedback comment received from the Market Agent.

  $$H = \{<\mathrm{Prop}_t, \mathrm{Comment}_t>\}, \quad \mathrm{Comment}_t \in \{\mathrm{winner}, <\mathrm{Eval}_{t1}, \ldots, \mathrm{Eval}_{tn}>\} \quad (4)$$

  where:
  - *$Prop_t$* is the proposal sent during the negotiation round *t*.
  - *$Comment_t$* is the comment received from Market Agent to proposal *$Prop_t$*. This comment indicates if the proposal is either the winner in the current round (*winner*) or includes a qualitative appreciation for each one of the attribute values under negotiation.
- The *Q* parameter includes relevant information to be used by the learning algorithm used for next bid formulation. This particular topic is discussed in a later section. At this point, it is only important to say that *Q* is described as follows:

  $$Q = \{Q_t\}, \quad Qt = \{<\mathrm{State}_t, \mathrm{Action}_t, \mathrm{QValue}_t>\}$$

3.2 Multi-Attribute Bid Evaluation

Negotiation implies, for the VO scenario as well as for most of the economic transactions, to take into consideration not only one, but multiple attributes for defining the terms (goods/services) under discussion. For instance, although the price of any good is an important (perhaps the most important) attribute, about delivery time or quality can also be, and generally are, complementary issues to include in the decision about to buy/sell or not a specific good.

Attaching utility values to different attributes under negotiation solves the problem of multi-attribute evaluation. Generally, an evaluation formula is a linear combination of the attributes' values weighted by their corresponding utility values. In this way, a multi-attribute negotiation is simply converted in a single attribute negotiation, where the result of the evaluation function can be seen as this single issue. Examples of this method are presented in [1, 2].

However, in some cases, it could be difficult to specify absolute numeric values to quantify the attributes' utility. A more natural and realistic way is to simply impose a preference order over attributes. The multi-attribute function presented in formula (5) encodes the attributes' and attributes values' preferences in a qualitative way and, at the same time, accommodates attributes intra-dependencies.

$$Ev = \frac{1}{Deviation}, \quad Deviation = \frac{1}{n} * \sum_{i=1}^{n} \frac{i}{n} * dif(\,PrefV_i, V_i\,), \tag{5}$$

where:

n = number of attributes that defines a specific component,

$V_x = f(V_1, \ldots, V_n), \quad x \notin \{1, \ldots, n\}$[1], and

$$dif(\,PrefV_i, V_i\,) = \begin{cases} \dfrac{V_i - PrefV_i}{max_i - min_i} & , \ \textit{if continuous domain} \\ \dfrac{Pos(\,V_i\,) - Pos(\,PrefV_i\,)}{nvalues} & , \ \textit{if discrete domain} \end{cases}$$

A proposal's evaluation value is calculated by the Market Agent, as the inverse of the weighted sum of the differences between the optimal (*PrefV*$_i$) and the real (V_i) value of each one of the attributes. In the formula, each parcel should be presented in increasing order of preference, that is, attributes identified by lower indexes are least important than attributes identified with higher indexes. The proposal with the highest evaluation value so far is the winner, since it is the one that contains the attributes' values more closely related to the optimal ones from the Market Agent point of view.

The negotiation process is realized as a set of rounds where Organisation Agents concede, from round to round, a little bit more trying to approach the Market Agent preferences, in order to be selected as partners of the VO. The Market Agent helps Organisation Agents in their task of formulating new proposals by giving them some hints about the direction they should follow in their negotiation space. These hints are given, by the Market Agent, as comments about attributes' values included in current proposals.

Qualitative Feedback Formulation

The response to proposed bids is formulated by the Market Agent as a qualitative feedback, which reflects the distance between the values indicated in a specific proposal and the optimal one received so far. The reason why the Market Agent compares a particular proposal with, not its optimal, but the best one received so far, can be explained by the fact that it is more convincing to say to an Organisation Agent that there is a better proposal in the market than saying that its proposal is not the optimal one.

A qualitative feedback is then formulated by the Market Agent as a qualitative comment on each of the proposal' attributes values, which can be classified in one of three categories: *sufficient*, *bad* or *very_bad*.

Organisation Agents will use this feedback information to its past proposals, in order to formulate, in the next negotiation rounds, new proposals trying to follow the hints included in the feedback comments.

[1] Each attribute V_x may depend on several other attributes through function *f*.

3.3 Learning in Bid Formulation

The Q-Negotiation algorithm uses a reinforcement learning strategy based in Q-learning for the formulation of new proposals. The Q-learning algorithm [6] is a well known reinforcement learning algorithm that maps evaluation values (Q-values) to pairs state/action.

The selection of a reinforcement learning algorithm seems to be appropriate in the negotiation process that conduits to the VO formation, since organization agents evolve in an, at least, partially unknown environment. And in particular, Q-learning enables on-line learning, which is an important capability in our specific scenario where agents will learn in a continuous way during all the negotiation process, with information extracted from each one of the negotiation rounds, and not only in the end with the negotiation result.

Q-learning is based in the idea of rewarding actions that produces good results, and punishing those that produce bad results, as indicated by parameter *r* in the correspondent formula (see equation (6)).

$$Q(s,a) \quad = \quad Q(s,a) + \alpha\left(r + \gamma \max_{b} Q(s',b) - Q(s,a) \right) \tag{6}$$

In the Q-Negotiation process, we assume that:

- A state is defined by a set of attributes' values, thus representing a proposal.

$$s = \langle v_1, v_2, \ldots, v_n \rangle \qquad \begin{array}{l} ,\ n = number\ of\ attributes \\ ,\ v_x\ :value\ of\ attribute\ x \end{array}$$

- An action is a relationship that is a modification of the attributes' values through the application of one of the functions: increase, decrease, or maintain.

$$a = \langle f_1, f_2, \ldots, f_n \rangle \qquad \begin{array}{l} ,\ n = number\ of\ attributes \\ ,\ f_x \in \{increase, decrease, maintain\} \end{array}$$

The adaptation of the Q-learning algorithm to our specific scenario, the VO formation negotiation, leads to the inclusion of two important features we will briefly enumerate in next paragraphs, and are detailed elsewhere [6].

The reward value for a particular state is calculated according to the qualitative feedback received from the Market Agent, in response to the proposal derived from this state (see formula 7).

$$r \quad = \quad \begin{cases} n & ,\ \ if\ \ winner \\ \dfrac{n}{2} - \sum\limits_{i} penalty_i & ,\ \ if\ \ not winner \quad \left(0 \le penalty_i \le 1\right) \end{cases} \tag{7}$$

The exploration space, which can became very large and thus implies a long time to learn, is reduced in order to include only those actions that can be considered as promising actions. A promising action is an action that can be applied to a previous state proposed to the Market Agent hints included in the feedback formulated by this agent. As an example, if the Market Agent, as a proposal's feedback, classifies the value of attribute *x* as *bad*, one promising action should be increase a little bit this attribute and maintain all the others.

3.4 Distributed Dependencies Resolution

One of the requirements for the negotiation protocol we are here proposing, besides dealing with attributes intra-dependencies, is the capability to deal with attributes' inter-dependencies. This is an important requirement to be considered in our scenario, because in the VO formation process interdependent negotiations take place simultaneously, and proposals received from different organisation agents may have incompatible dependent attributes' values. Therefore, agents should negotiate in order to agree between them on mutual admissible values, what can be seen as a distributed dependencies satisfaction problem.

The distributed dependencies satisfaction problem has been the subject of attention of other researchers, addressing the study of both single [7] and multiple dependent variables [8, 9, 10]. In the VO formation process, dependencies may occur between multiple variables, making the latter approaches more relevant to our research. The first two mentioned papers, [8, 9] describe algorithms to reach one possible solution, not the optimal one. The third paper [10] introduces an algorithm that, although reaching the optimal solution, imposes that all agents involved in the mutual dependencies resolution process have to know all agents' private utility functions.

Differently from all these proposals, our distributed dependencies satisfaction algorithm, besides reaching the optimal solution, keeps agents' information as much as possible private.

Each agent involved in the distributed dependent problem resolution should know its space of states, that is, all possible values for its own dependent attributes. Agents will then exchange between them alternative values for the dependent attributes, in order to approach an agreement. As in any iterative negotiation process, agents start the negotiation by proposing its optimal (from a local point of view) solution and, in the next rounds start conceding trying to reach a consensus.

In order to properly understand the way the algorithm works, first we should introduce the concept of *"decrement of the maximum utility"* of an alternative state. State transitions are due to relaxation of one or more state variables. The decrement of the maximum utility of a particular alternative proposal can be calculated as the difference between the evaluation values of this alternative proposal and the optimal one. We will abbreviate "decrement of the maximum utility" to "decrement of the utility" meaning the successive amount of utility agents has to concede compared to the (local) optimal bid. Formula (8) represents the decrement of utility for agent *i*, corresponding to the particular state s^k, where s^* is the agent's optimal state (proposal).

$$du_i^k = Ev\left(s^*\right) - Ev\left(s^k\right) \tag{8}$$

At each negotiation step, the agent selects as a new proposal the one that has the lowest decrement of the utility of those not yet proposed. During the negotiation process, agents do not reveal their own state's utility, but only the state's decrement utility, what enables keeping important information private.

This process ends when all agents cannot select a next state better than one already proposed in the past. In this way, agents, although remaining self-interested, will converge for a solution that is the best possible for all of them together, because it represents the minimum joint of decrement of the utility.

The proposed distributed dependencies satisfaction algorithm can be described as follows:

1. Each agent *i* select its next preferable alternative state, from those not yet proposed before. Let us suppose this is state *a.*

   $$du_i^a = \min_s \left(du_i^s \right)$$

2. Each agent *i* sends out to others:
   - its own preferable state as a new proposal
   - its own local decrement of the utility for that state
3. When agent *j* receives the proposal (state a) from agent *i*, it calculates:
   - its own local decrement of utility (du_j^a)
   - the joint decrement of the utility:

     $$jdu^a = \sum_{dag} du_{dag}^a \quad , dag = \{1 \ldots n\} \textit{ set of mutual dependent agents}$$

   - the minimum joint of the decrement of the utility already known (jdu^m)
4. and selects:
   - its next preferable state. Suppose it is state *b*.
   - if $du_j^b < jdu^m$, agent *j* proposes state b to other agents
   - else agent *j* accepts state *m* as the final proposal and negotiation ends.

Transfer of Compensations

After agree in a global solution, agents involved in the dependencies resolution process, generally get different local decrement of utility values and, therefore, some agents become more penalized than others. In order to guarantee that all agents involved in the distributed dependencies resolution get the same real decrement of utility (*rdu*), the joint decrement of the utility will be distributed between them according to formula (9):

$$rdu = \frac{jdu^m}{n}, \quad n = number of agents \tag{9}$$

As a consequence, some agents have to pay or get a compensation value to others. Once agent *i* has previously calculated du_i^m as its local decrement of utility, the compensation value is calculated according to formula (10).

$$cValue_i = rdu - du_i^m \tag{10}$$

If the agent' real decrement of the utility is greater than its local decrement of the utility, it will pay a compensation value to others, that has calculated as the difference of these two values. If not, the agent will get a compensation value.

4 Phased Commitments

We have seen how agents, representing individual autonomous enterprises, may reach an agreement through appropriate negotiation procedures. The contract that formalises that agreement should explicitly state all the commitments that those agents are due to satisfy all along the VO life cycle.

A full commitment contract may be unable to deal with possible future, partially unexpected, events. This fact has already been recognized since the definition of the old contract net protocol [11] where the possibility of a contract cancellation was envisaged. More recently, other authors like [12] have approached this subject in the context of decommiting in the meeting scheduling application. However, it was Sandholm [13, 14] who gave a more systematic and relevant contribution for this issue through the introduction of the concept of "leveled commitment" and associated penalties. Contrary to the game theoretic approach where contingency contracts are established according to the existence or not of future events, Sandholm [14] allows unilateral decommitments through the payment of calculated penalties. Resulting contracts are then called "leveled commitment contracts".

Three main aspects are related to this issue:

- First, the problem of how to represent, in an unambiguous form, such a commitment including all relevant information about future agents' attitudes.
- Second, how to explore this knowledge in order to correctly monitoring the next stages of the VO life cycle?
- Third, what to do in case of failure of what was previously negotiated and agreed and finally stated in the accepted contract. Can parties back out? In what circumstances can that happen? And, if the answer is yes, what procedure should follow? Should a new negotiation process start or (and) should appropriate penalties be enforced on the agents?

Commitments have to be agreed and represented in such a way that at several different future points in time, they can be verified. Which procedure to follow after that verification should also be considered in the agreed contract.

We envisage representing the negotiation outcome (the agreed contract) as a kind of frame where each slot represents pre-conditions plus a set of rules to be selected for possible application. Those rules whose conditional part has been verified indicate the appropriate action to be taken in those specific circumstances.

A contract, including a set of phased commitments, can be represented as:

$$Contract = \left\langle \ LAgts, Agts - P/S, Verification\_procedure \right\rangle \text{, where:}$$

- *LAgts* represents the list of agents that accept that contract.
- *Agt-P/S* ties each agent together with the contribution (product or service) is commited to give to the VO.

$$Agt - P/S = \{Agt - P/S_i\}, \quad Agt - P/S_i = \left\{ \left\langle Agt_i, Prod/Serv_i \right\rangle \right\}$$

- *Verification_procedure* indicates how and when to monitoring the operation procedures agreed through the contract. *Verification_procedure* is represented as:

Verification_procedure=<Pre-cond, Rule-set>, where:

- *Pre-cond* ∈ {*event*, *time_point*}
  - *event* is a specific type of arrived messages.
  - *time _point* is a pre-specified point in time for checking current conditions.
- *Rule-set = {<cond$_i$, action$_i$>}*
  - *cond$_i$* is a set of conditions to be checked after *Pre-cond* is true.
  - *action$_i$ ∈ {dpenalty, dpenalty + re_negt, noact}*, and:
    - *dpenalty* represents a decommitment penalty value.
    - *re_negt* represents the re-negotiation action.
    - *noact* represents the case where no action is to be done.

Decommitment penalties have to be calculated according to the other VO partners' respective losses. It is our intention to enhance the negotiation protocol in such a way that the agents already know these penalties at the end of the negotiation phase.

5 Conclusions and Discussion

Electronic Institutions are general frameworks for helping in collaborative work in electronic environments. Electronic Institutions provide, sometimes enforce, rules and norms of behaviour and make available service facilities supporting both interaction and operation monitoring of computational entities.

A Virtual Organization is a powerful example of the need of such collaborative work, once different enterprises have to join together, temporarily, to achieve a common business oriented goal. This paper elaborates on how Electronic Institutions can effectively help during the VO life cycle.

For the VO formation stage we have introduced a new negotiation algorithm, called Q-Negotiation, which includes appropriate features for dealing with the specific requirements of the VO scenario. An important requirement in the VO scenario, is that information must be kept private to individual enterprises, since they are competitive by nature and do not want to reveal their market strategy to others. The Q-Negotiation algorithm has the ability to maintain information private to individual enterprises, and at the same time, includes the capability to evaluate multi-attribute proposals, to learn during the negotiation process, and to resolve attributes' inter dependencies. Let us discuss each one of these features separately. First, multi-attribute evaluation is done assigning relative preferences to attributes. Other studies in multi-attribute evaluation [1, 2] generally impose the use of a real concrete value that captures each attribute's importance, which sometimes can be difficult to quantify. Second, learning is performed by using an on-line reinforcement learning algorithm during the negotiation process, through a qualitative feedback that is the opponent's comment to each proposal. Third, the inter attributes' dependencies resolution process proposed in Q-Negotiation reach the optimal solution keeping information private as much as possible. Other known approaches related with distributed dependencies resolution, either reach non-optimal solutions [8, 9], or impose the knowledge of other agents' private information to be made public [10].

For the VO operation stage we here propose the exploration of a phased commitment that is established in the end of the negotiation process during VO life cycle previous stage. This commitment is then specified in a contract. Through phased commitments, the Electronic Institution has the capability to monitor the behaviour of the participant entities at pre-specified moments previewed in the contract according to time points or future events.

We intend to develop in the future knowledge representation and ontology [15] services appropriate for Electronic Institutions.

References

1. Vulkan, N., Jennings, N.R.: Efficient Mechanisms for the Supply of Services in Multi-Agent Environments. 1st Int Conf. on Information and Computation Economics, Charleston (1998)
2. Matos, S., Sierra, C.: Evolutionary Computing and Negotiation Agents. Workshop on Agent Mediated Electronic Trading, Minneapolis, USA (1998)
3. Faisst, W.: Information Technology as an Enabler of Virtual Enterprises: A Life-cycle-oriented Description. European Conference on Virtual Enterprises and Network Solutions, Paderborn, Germany (1997)
4. Fischer, K., Muller, J.P., Heimig, I., Scheer, A.: Intelligent Agents in Virtual Enterprises. 1st International Conference on the Pratical Application of Intelligent Agents and Multi-Agent Technology, London, UK (1996)
5. Watkins, C.J.C.H., Dayan, P.: Q-Learning. Machine Learning, Vol. 8, N. 3/4. Kluwer Academic Publishers, Boston (1992) 279-292
6. Oliveira, E., Rocha, A.P.: Agents advanced features for negotiation in Electronic Commerce and Virtual Organisations formation process. In: Dignum, F., Sierra, C. (eds.): Agent Mediated Electronic Commerce, the European AgentLink Perspective. Lectures Notes in Artificial Intelligence, Vol. 1991. Springer-Verlag (2000) 77-96
7. Yokoo, M., Durfee, E., Ishida, T., Kuwabara, K.: Distributed Constraint Satisfaction for Formalizing Distributed Problem Solving. 12th International Conference on Distributed Computing Systems, Yokohama, Japan (1992)
8. Armstrong, A., Durfee, E.: Dynamic Prioritization of Complex Agents in Distributed Constraint Satisfaction Problems. 15th International Joint Conference on Artificial Intelligence, Nagoya, Japan (1997)
9. Yokoo, M., Hirayama, K.: Distributed Constraint Satisfaction Algorithm for Complex Local Problems. 3rd International Conference on Multi-Agent Systems, Paris, France (1998)
10. Parunak, H.V.D., Ward, A., Sauter, J.: The MarCon Algorithm: A Systematic Market Approach to Distributed Constraint Problems. Artificial Intelligence for Engineering Design, Analysis and Manufacturing J., Vol. 13. Cambridge Univ. Press (1999) 217-234
11. Smith, R.G.: The Contract Net Protocol: High-Level Communication and Control in a Distributed Problem Solver. IEEE Trans. on Computers, Vol.29, N.12 (1980) 1104-1113
12. Sen, S., Durfee, E.: The role of commitment in cooperative negotiation. International Journal on Intelligent Cooperative Information Systems, Vol.3, N.1 (1994) 67-81
13. Sandholm, T.W., Lesser, V.R.: Advantages of a Leveled Commitment Contracting Protocol. In Proceedings of the Nat. Conf. on Artificial Intelligence (AAAI), Portland (1996) 126-133
14. Sandholm, T.: Agents in Electronic Commerce: Component Technologies for Automated Negotiation and Coalition Formation. Autonomous Agents and Multi-Agent Systems, Vol,3, N.1. Kluwer Academic Publishers (2000) 73-96
15. Gruber, T.R.: A Translation Approach to Portable Ontology Specifications. Knowledge Acquisition, Vol.5, N.2, (1993) 199-220

An Imitation-Based Approach to Modeling Homogenous Agents Societies

Goran Trajkovski

West Virginia University - P, 300 Campus Drive, Parkersburg, WV, 26104, USA
E-mail: Goran.Trajkovski@mail.wvu.edu

Abstract. The present paper concentrates on one modeling approach for homogenous societies of agents in a given environment. It is an extension of the existing Interactivist-Expectative Theory on Agency and Learning to multi-agent environments. The uniagent theory's key phrases are expectancy and learning through interactions with the environment. Motivated by the research done in the domain of imitation in humans, this paper introduces learning by imitation through interaction between akin agents. The social consequences of such an environment from the perspective of learning and emergence of language are discussed as well.

1 Introduction

In our work on within the Interactivist-Expectative Theory of Agency and Learning (IETAL) [8], we proposed and investigated a learning paradigm in autonomous agents in a uniagent environment. The key concepts of this theory are expectancy and learning the environment through interactions, while building an intrinsic model of it. Depending on the set of active drives, the agent uses the appropriate projection of its intrinsic model in order to navigate within the environment on its quest to satisfy the set of active drives.

In this paper we take off with the existing results of the theory and propose the basis of a multiagent theory, where apart from the notions of expectancy and learning through interactions with the environment, we introduce interaction, [2], between alike agents, inspired by research results on the phenomenon of imitation [1], [4], in both neurophysiology and psychology.

The paper is organized as follows. Section 2 summarizes some of the relevant results in research of the phenomenon of imitation in humans. In Section 3, we give the introduction to the IETAL Theory. In Section 4, we formalize an instantiation of the IETAL theory in the agent that we call Petitagé, which interacts with the environment it is in, and builds its intrinsic model of it. By equipping the agent with a special sensor for sensing the alike agents, and enabling the agents to exchange each other's contingency tables, we allow the unit in an environment inhabited with Petitagé-like agents to function as a homogenous multiagent world with socially active agents. This environment and the emergence of language is the focus of discussion in Section 5. The last section overviews the paper, and states directions for further investigation in the domain.

2 Imitation Revisited

In this section we summarize relevant results from research from neurophysiology and psychology applicable in the domain of imitation, in our approach towards the modeling of the multiagent society. This approach is justifiable by the efforts towards a more efficient learning paradigm in agents with the embodied mind paradigm.

The concept of learning by imitation (learning by watching, , teaching by showing) is certainly not a new one. Thorndike [10] defines it as “[imitation is] learning to do an act from seeing it done, whereas Piaget [5] mentioned as a major part when offering his theory for development of the ability to imitate going through six stages. After Piaget the interest in (movement) imitation diminished, partially because of the prejudice that “mimicking” or “imitating” is not an expression of higher intelligence, [7]. Imitation is, though, far from being a trivial task. It is rather a highly creative mapping between the actions and their consequences of the other and of one-self’s (visual input-motor commands-consequences evaluation). Rizzolatti et al, [6], discover the so called “mirror neurons” - neurons that fire while one is performing some motor actions or look at somebody else doing the same action.

This finding of Rizzolatti may give insights in our empathizing abilities as well as our “other mind reading” abilities. It states that in the addition of the first person point of view (subjective experience) and the third person point of view (scientific, or objective stance) we have a special within-the-species shared point of view dedicated on understanding others of the same species. Consequences for cognitive science and AI include putting radical constraints within the space of possible models of the mind

Within the classical, disembodied approach recognizing and communicating with *others* was not a real problem. New knowledge (i.e. new combinations of symbols) was easily communicated via inherently linguistic terms, due to the assumption that everybody and everything have access to some internal mirror representation, [3].

3 Basics of the UniAgent Theory

In this section we summarize the uniagent IETAL theory in order to facilitate the communication of ideas in the remaining part of the paper.

The notion of expectancy has a central role in this theory, and the agent, while being in the environment, anticipates the effects of its own actions in the world. In order to avoid any possible terminological confusion, we give the following explanation. An agent is said to be *aware* of the environment it inhabits if it can *anticipate* the results of his own actions. This means that, given some current percept p the agent can generate expectancies about the resulting percepts p_1, p_2,... if it applies actions a_1, a_2... After inhabiting some environment for certain time, an agent builds a network of such expectancy triplets p_i-a_j-p_k. This brings us to the second key concept in our theory of agency, namely the concept of agent environment interaction**.** As we see, the main problems are: first, to learn the graph and second, to know how to use it.

4 Formalization of the Agent

In this Section we give an algebraic formalization of the agent that inhabits the multiagent environment, [11]. Formalizing the agent and its interaction(s) is crucial for the experimental setups in the exploration of the efficiency of the approach. The agent is gaining its knowledge throughout its stay in the environment.

If we speak in terms of graphs, we may depict this situation with a graph whose nodes are labeled and these labels may be same for different nodes. The only way a node can be recognized as different then is to examine its context. Context of a node is a tree-like data structure defined recursively as all followers of the node together with their contexts. In [11], we have presented learning algorithms inspired by biological systems.

Let $V=\{v_1, v_2, ..., v_n\}$be a finite nonempty set of vertices and $A=\{s_1,s_2, ... , s_m\}$a finite nonempty set of actions with implicit ordering induced by the indexing. Let the pair Gs = (V, r_s), $s\in A$, where $r_s\subseteq V\times V$,be oriented regular graph with matrix of incidence of the relation r_s for every $s\in A$ that contains exactly one element 1 in each column.

The graph G' = $(V, \bigcup_{s\in V} r_s)$ will be called *graph of general connectivity* of the family of graphs $\{G_s : s\in A\}$. If the graph G' is connected, then the triplet G=(V, A, r), $r\subseteq(V\times V)\times A$, defined as follows $((v_1, v_2),s)\in r$ if and only if $(v_1,v_2)\in r_s$ $v_1,v_2\in V$, $s\in A$, represents an *oriented graph with marked edges.* V is set of vertices, r relation, and A set of actions of G..

Let $L=\{l_1, l_2, ..., l_k\}$be a set of labels (and $f : V\to L$ a surjection. F is called labeling of the vertex set of G, and it endorses the possibility of modeling the perceptual aliasing.

The graph that we, as designers of the experimental setup, see G''=(V, A, L, r, f) is is the Designer Visible Environment, and the graph G''' = (L, r''', A) is the Agent Visible Environment (model). This is what the agent is able to “see”.

During its stay in the environment, the agent is interacting with the environment, and builds its intrinsic representation of the environment (Fig 1).

When the, say, hunger drive is activated for the first time, the agent performs random walk during which expectancies are stored in the associative memory. The emotional contexts of these expectancies are neutral until food is sensed. Once this happens, current expectancy emotional context is set to positive value. This value is then exponentially decremented and propagated backwards following the recent percepts.

Every next future time the hunger drive is activated, the agent uses the context values of the expectancies to direct its actions. It chooses, of course, the action that will lead to expectancy with maximum context value. If all the expectancies for the current perceptual state are neutral, random walk is performed. Again, when food is sensed emotional contexts are adjusted in the previously described manner.

```
Generate_Intrinsic_Representation (G: Interaction_Graph,
    ξ: Schema; GIR: Assotiative_Memory)
```

```
BEGIN_PROCEDURE
    Initialize ($R_\Delta$=∅); Initialize_Position (G; Position);
    Try (Position, ξ; ($B_1$, $S_1$));
    Add ( [(λ, λ), ($B_1$,$S_1$)]; $R_\Delta$);
    WHILE (Active_Drive_Not_Satisfied) DO
            Try (Position, ξ; ($B_2$, $S_2$));
            Add ([($B_1$, $S_1$), ($B_2$, $S_2$)]; $R_\Delta$);
            ($B_1$, $S_1$):=($B_2$, $S_2$);
    END_WHILE
    Propagate_Context ( ($B_1$, $S_1$), drive; GIR).
END_PROCEDURE

Try (Position: Location_In_Interaction_Graph, ξ: Schema;
    (B, S): Percepts_Actions_Pair)
BEGIN_PROCEDURE
    S:=λ;
    TryIn (Position, ξ; (Add (S, Current_Percept), B));
    REPEAT
            TryIn (Position, B; (Add (S, Current_Percept), B)
    UNTIL NOT enabled (B)
END_PROCEDURE

Propagate_Context (d: drive; GIR: Assotiative_Memory)
BEGIN_PROCEDURE
    N:=0;
    WHILE ($B_1$, $S_1$) ∈Projection$_2$ (GIR) DO
            Projection$_3$(GIR) := exp(-N);
            INC (N)
    END_WHILE
END_PROCEDURE
```

Fig. 1. A possible implementation of the learning procedure for the autonomous agent.

The way the agent "perceives" the environment is different when different drives are active. That means that it uses different instantiations, depending on the given set of drives. It is realistic to assume that the drives are ordered in a complete sub-semi-lattice manner (meaning that every subset of drives has its maximum within the structure of drives).

The top of the drive structure in all agents in the multiagent society is the drive to imitate the agents "in sight". Once two agents sense each other, they shift to the imitation mode, in which they exchange their contingency tables. The discussions on the consequences of the inter-agent interaction are discussed in the following section.

5 The Multiagent Society

In this Section we give a description of a multiagent environment inhabited by Petitagé-like agents, that are able to imitate their cohabitants. The problems of intraagent communication in heterogeneous multiagent environments are discussed at the end of the Section.

As discussed before, the agent is equipped with a special sensor that is sensing other Petitagé-like agents in the same environment. The sensor is working in parallel with the other sensors. Sensing another Petitagé takes the agent into its imitating mode, during which the agents exchange their contingency tables. The tables are being updated by new rows.

Let us suppose that the contingency table of one of the agent has *M* rows, and of the other *N* rows. After the process of exchange of the contingency table, both of the agents update their tables, with the union of their individual tables. In mathematical terms, that would mean that the cardinality of the relations (number of rows of the contingency table) will be at most $M+N$.

The agents might have been visiting different areas of the environment during their walk through the environment, or areas that are perceptually very similar to each other, which they have inhabited for durations of time T_1, and T_2 respectively. Depending on the topology of those areas of the environment, and their similarity, the intersection of the relations can range between *0* and min $\{M, N\}$, the first corresponding to a visitation to perceptually different environments, and the latter to visitation of perceptually very similar environments.

The results form the uniagent IETAL cannot be generalized to state that the longer an agent has been in the environment, the “longer” the contingency table will be. Therefore, the “age difference” abs $(T_1\text{-}T_2)$ has no influence on the shapes on the learning curves. On the other hand, the learning curve decreases in time, [8]. However, due to the solution of the problem of cycling in a simulated environment by introduction of random moves once cycling is detected, the odds of the agent to explore a more comprehensive part of the whole environment are increased. Nevertheless, after the process of information interchange, the learning curve drops, and is in the band between the time axis and the lower of the individual learning curves of each of the agents.

Upon the first look the society of Petitagés would be easily ordered by the amount of information contained in the contingency tables. However, due to the history of the agent and the part of the environment visited since, the agents in the environment cannot be ordered in a linear manner by inclusion of the contingency tables.

Algebraically, the structure with a carrier all possible contingency tables in a given environment, ordered by set inclusion, from the designer’s point of view, is a lattice that we refer to as the lattice P of all Petitagés. The bottom element an agent with no knowledge of the environment, which has been made inhabitant of the environment, and top an agent that has been in the environment long enough to know everything. The agents will clusterize in sublattices of agents that have been visiting similar areas of the environment. As agents come into the environment and start learning, they walk through P, too, moving in the direction of the top element. All the agents in the environment at a given time t, consist a structure S(t), that is a general partially ordered set by inclusion, with both comparable and incomparable elements. In other terms, as a graph it may not be connected at a given point in time.

Every element of the substructure of the hierarchy rooted at a given agent A, knows what A knows at l east what A knows. Its contingency table is subset of the tables of the agents comparable and higher to A in P. The contingency table of the root of the class of agents rooted at A at in P will be referred as to an **entry** in their **lexicon**. After an imitation instance, their lexicon will naturally change by expanding the lexicon.

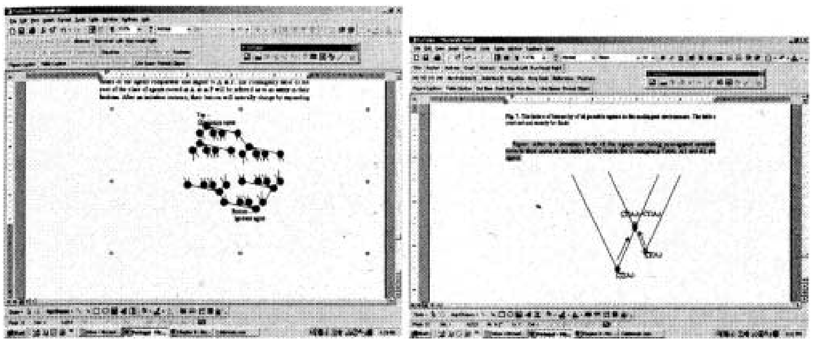


Fig. 2. The lattice of hierarchy of all possible agents (left). After the imitation, both agents involved in the contingency table interchange, are propagated upwards in the lattice P. CT stands for contingency table, A_1 and A_2 are agents.

The lexicon should be understood as a system of activities that all the agents in the class can perform. The language of the AA society will consist of the union of lexicographic entries of all agents currently in the environment.

No general interrelation can be stated between the knowledge and the age hierarchies of the agents in the world of Petitagés. As stated before, no guarantee can be given that the longer an agent inhabits the environment, the more it "knows". The same statement holds even for agents with the same lexicons, i.e. belonging to a class rooted at the same agent in P or S(t).

The problem of inability to exchange the whole contingency tables due to hardware or time constraints is a practical constraint that needs to be taken into serious considerations while simulating a multiagent environment inhabited by Petitagé-like agents.

The problem off lack of time for convention (the other agent got out of sight before the contingency tables were exchanged) is not as severe as the problem of hardware memory constraints. The imitation stage pulls both agents higher in P; they actually gain more knowledge of the environment. While addressing the memory problem, policies of forgetting need to be devised.

6 Conclusions and Directions for Further Work

In this paper we proposed a model for a multiagent society, based on expectancy and interaction. Based on results on imitations, we proposed a multiagent version of the IETAL theory. The Petitagé-like agents inhabit the same environment and interact with each other using imitation. The drive to imitate is the highest on the hierarchy of drives of all agents. After the convention of two agents, they interchange their contingency tables, and continue to explore the environment. The problems associated with the imitation stages and the changes in the agents have been observed from different angles.

The abundance of approaches towards multiagent systems that are not necessarily compliant with the mainstream AI, give us the motivation to explore further the theory. In an upcoming stage of our research, we will simulate the World of Petitagés, and explore the problems that such a simulation will reveal, for a later implementation of the approach in robotic agents.

References

[1] Byrne R W, Russon A E, " Learning by Imitation", Behavioral and Brain Sciences (2001).
[2] Ferber J: Multi-Agent Systems: An Introduction to Distributed Artificial Intelligence, Addison-Wesley, 1999.
[3] Fikes R E, Nilsson N J, "Learning and executing generalized robot plans", Artificial Intelligence, 3(4) (1972).
[4] Nehaniv C, Dautenhahn K, "Mapping between dissimilar bodies: Affordances and the algebraic foundations of imitation", Proceedings of the Seventh European Workshop on Learning Robots Edinburgh , UK (1998) 64--72.
[5] Piaget, J Play, Dreams, and Imitation in Childhood, Norton, New York, 1945.
[6] Rizzolatti G, Fadiga L, Gallese V, Fogassi L, Premotor cortex and the recognition of motor actions. Cognitive Brain Research, 3(2) (1996) 131-141
[7] Schaal S, Sternad D, "Programmable pattern generators", Proceedings, 3rd International Conference on Computational Intelligence in Neuroscience", Research Triangle Park, NC, (1998) 48-51.
[8] Stojanov G, Bozinovski S, Trajkovski G, "Interactionist-Expectative View on Agency and Learning", IMACS Journal for Mathematics and Computers in Simulation, North-Holland Publishers, Amsterdam, Vol. 44 (1997).
[9] Stojanov G, Trajkovski G, Bozinovski S, "The Status of Representation in Behavior Based Robotic Systems: The Problem and A Solution", IEEE Conference Systems, Man, and Cybernetics, Orlando (1997).
[10] Thorndike E L, "Animal intelligence: an experimental study of the associative process in animals" Psychological Review Monograph 2(8) (1898) 551-553.
[11] Trajkovski G, Stojanov G, "Algebraic Formalization of Environment Representation", in: (Tatai, G., Gulyas, L. (eds)) Agents Everywhere, Springer, Budapest, HU (1998) 59-65.

Situation Calculus as Hybrid Logic: First Steps*

Patrick Blackburn[1], Jaap Kamps[2], and Maarten Marx[2]

[1] INRIA, Lorraine, Nancy, France. patrick@aplog.org
[2] ILLC, University of Amsterdam, Plantage Muidergracht 24, 1018TV Amsterdam, the Netherlands. {kamps,marx}@science.uva.nl

Abstract. The situation calculus, originally conceived by John McCarthy, is one of the main representation languages in artificial intelligence. The original papers introducing the situation calculus also highlight the connection between the fields of artificial intelligence and philosophical logic (especially modal logics of belief, knowledge, and tense). Modal logic changed enormously since the 60s. This paper sets out to revive the connection between situation calculus and modal logic. In particular, we will show that quantified hybrid logic, *QHL*, is able to express situation calculus formulas often more natural and concise than the original formulations. The main contribution of this paper is a new quantified hybrid logic with temporal operators and action modalities, tailor-made for expressing the fluents of situation calculus.

1 Introduction

The seminal paper that McCarthy and Hayes published in 1969, *Some Philosophical Problems from the Standpoint of Artificial Intelligence*, marks a watershed in artificial intelligence. It is the key reference for one of its main representation languages—the situation calculus. We will focus here on the original version of situation calculus ([13,14]; sometimes called the "snapshots" version, to distinguish it from other variants). The most important construct of situation calculus is—no surprise—situations. As [13] has it:

> One of the basic entities in our theory is the *situation*. Intuitively, a situation is the complete state of affairs at some instant of time. ... Since a situation is defined as a complete state of affairs, we can never describe a situation fully; and we therefore provide no notation for doing so in our theory. Instead, we state facts about situations in the language of an extended predicate calculus. Examples of such facts are 1. raining(s) meaning that it is raining in situation s.

The situations are fully informed instances of the world of which we have limited knowledge, but still occur in the object language—this is what modal logicians now call a hybrid language. Precisely the same intuition is present in the writings of Arthur Prior, the founder of temporal logic [17]. McCarthy and Hayes [14] praise Prior's work. They include his temporal operators into the situation calculus and they note the similarity

* This research was supported by the Netherlands Organization for Scientific Research (NWO, grants # 400-20-036 and # 612-62-001). This work was carried out as part of the INRIA funded partnership between CALG (Computational and Applied Logic Group, University of Amsterdam) and LED (Langue et Dialogue, LORIA, Nancy).

of their use of situation variables to Prior's time-instants. But that is it. Apart from this promising beginning, the languages of situation calculus and the modal languages based on Kripke's and Prior's work have always stayed far removed from each other.

We think this is at least partly due to historical reasons. First of all, Prior's writing is notoriously difficult. Secondly, in the late 60's first order modal logic was a hot topic but the debate centered around all its philosophical problems. At that time hardly anyone saw it as a useful language for doing knowledge representation, with McCarthy and Hayes as notable exceptions. In fact, Prior is an exception too; he saw that modal logic could be used for a general (dynamic) theory of information. Another important reason was the inadequate expressive power of the available modal languages for the purposes McCarthy and Hayes had in mind. Since the late 60's, this situation has changed considerably. First and foremost, we know now that actions can be naturally represented in dynamic logic, a branch of modal logic.[1] Secondly, nowadays modal logic has become a respectable member in the field of knowledge representation, be it under the name of description logic.[2] Finally, the 90's saw the emergence of a branch of modal logic called hybrid logic which took up, or sometimes reinvented, many of Prior's ideas. E.g., seemingly unaware of Prior, Passy and Tinchev [15] argue for the introduction of names for states in dynamic logic. Hybrid logic adds to modal logic explicit reference to states, a mechanism to bind variables to states (the modal–logical term for situation), and a holds operator $@_i\phi$, allowing one to express that a formula ϕ holds *at* a state named i.

The purpose of this paper is to introduce hybrid logic to the artificial intelligence community. We will do this by showing that hybrid logic is very well suited to express what is normally formulated in the situation calculus. We have chosen for a comparison with the very first situation calculus language, from [14]. Our prime reason for choosing [14], apart from the fact that it started the field, is that one can feel their struggle with the first order language they are using. They have to introduce λ–abstraction, and all the time they introduce abbreviations to make their formulas look intuitive. These abbreviations foreshadowed a number of later technical developments in modal logic (e.g., van Benthem's celebrated standard translation into first order logic). In fact, we see McCarthy and Hayes as forerunners of the use of modal logic as a knowledge representation language and would not be surprized if they had used hybridized first order modal logic to state the situation calculus if only the right ingredients had been available when they wrote their article.

The rest of this paper is structured as follows. We start with with a brief introduction to hybrid logic. In the main part of the paper we show how to express typical situation calculus statements in hybrid logic. Here we gently introduce the notions of hybrid logic and show their use in examples. Rigorous definitions of its syntax and semantics are provided in the appendix. We end with a discussion of the presented work.

[1] Dynamic logic originates with V. Pratt [16]. The recent monograph [10] contains many applications of dynamic logic to computer science. The rendering of a version of the situation calculus in GOLOG by Levesque, Pirri and Reiter [12] is also based on dynamic logic.

[2] Description logic [5,8] evolved out of Brachman and Schmolze's knowledge representation language KL–ONE [6]. There are now a number of very fast DL provers for very expressive (exptime complete) languages, e.g., DLP and Racer, cf., the DL web page `http://dl.kr.org/`.

2 Hybrid Logic

The rapidly growing field of hybrid logic, although rooted in the philosophical logic of Prior, is now being recognized as a tool in the field of knowledge representation. Hybrid logic has close connections with the field of description logic (cf., the page `http://dl.kr.org/` or [8]). At present, several description logic theorem provers are being adjusted to handle the full nominals of hybrid logic. These provers handle propositional hybrid fragments with an exponential time worst case complexity with surprising efficiency. The proof and model theory of propositional hybrid logic is by now understood very well [3,2]. Recent unpublished work on first order hybrid logic indicates it has enormous advantages over first order modal logic. For instance, a complete analytic tableau system exists which also yields interpolants. One of the strong indications that something is missing in the usual formulation of first order modal logic is its failure of the interpolation property [9]. The computational and applied logic group at the University of Amsterdam is currently implementing a resolution–based theorem prover for hybrid logic. Carlos Areces maintains a web page devoted to hybrid logic at `http://www.hylo.net`. There have been a number of hybrid logic (HyLo) workshops. The next will be held as a LICS–affiliated workshop during the summer of 2002.

3 Situation Calculus as Hybrid Logic, First Steps

In this section we argue that hybrid logic is an excellently suited formalism to speak about situations and fluents. We do this by reviewing the key examples in [14] and reformulate them in hybrid logic. The hybrid language will be introduced informally and step by step. A rigorous formal definition of the resulting quantified hybrid logic can be found in the Appendix.

McCarthy and Hayes seem very much willing to suppress the situation argument in their formulas, just as in first order modal logic. This shows in all example formulas in section 2 of [14]. They find it unnatural (and going against natural language practice) to add an extra argument to each predicate symbol for the situation. For example "John loves Mary" has to be expressed as $love(j, m, s)$ where s refers to a situation. For this reason they introduce "abbreviations" in which this extra argument is suppressed. (We write this between quotes as the syntactical status of these formulas is not always clear.) Still they cannot do this in all cases because they sometimes need to refer to situations explicitly. They note the similarity with Prior's nominals:

> The use of situation variables is analogous to the use of time-instants in the calculi of world-states which Prior [17] calls U-T calculi. [14, p.480]

We will now show that the modern treatment of Prior's ideas which has become known under the name of *hybrid logic* provides exactly the linguistic elements that McCarthy and Hayes seemed to be searching for.

The two most important semantic constructs in the situation calculus are the *situation* and the *fluent*. A situation is the complete state of the universe at an instant of time. A fluent is a function whose domain is the set of situations. *Propositional fluents* are fluents whose range is the set of truth values $\{true,\ false\}$. *Situational fluents* are those whose range is the set of situations itself.

We start with considering propositional fluents. The key idea of situation calculus is that the meaning of every expression is a fluent. If we equate situations with the possible worlds from Kripke semantics, following the suggestion in [14, p.495], then sentences in quantified modal logic express propositional fluents. For example, the meaning of the sentence "John walks" is traditionally given as the set of possible worlds in which the sentence "John walks" is true. This set of course uniquely determines a propositional fluent.

Key idea of modal logic: Every first order modal logical sentence expresses a propositional fluent. It does so without referring explicitly to situations. In fact in traditional modal logic one can not refer to the situations (more traditionally called "worlds") in the models. Also in quantified hybrid logic (QHL) every sentence expresses a propositional fluent. But in addition one can refer to situations and indicate that a formula holds at a certain situation.

Names for situations and a holds operator. But McCarthy and Hayes need more expressive power than quantified modal logic has to offer. They want to be able to express "At situation s, 'John walks' holds".[3] This is not possible in quantified modal logic because it contains no machinery to refer to possible worlds.

This is where Prior's ideas and their modern treatment in the form of hybrid logic come into action. For the moment, add a second sort of variables, called *nominals*, to the language of first order logic. Every nominal is a formula, and nominals can be freely combined to form new formulas. In addition, whenever i is a nominal and ϕ is a formula, then also $@_i\phi$ (pronounce: at i, ϕ) is a formula.

The function of nominals is to *name* situations. The meaning of a nominal i—an atomic formula in hybrid logic—in a model will be the propositional fluent which is true only for the unique situation that is named by i in the model. $@_i\phi$ adds a holds–operator to first order logic: $@_i\phi$ states that the formula ϕ holds at the situation named i. Thus the meaning of $@_i\phi$ is the constant propositional fluent which sends every situation to *true* if ϕ holds at the situation named i, and every situation to false otherwise.

Let's consider the first example from [14, p.478]. McCarthy and Hayes want to "assert about a situation s that person p is in place x and that it is raining in place x." This is expressed by $at(p, x, s) \wedge raining(x, s)$. Not being satisfied with this notation they give two other possible equivalent notations:

$$[at(p, x) \wedge raining(x)](s) \tag{1}$$
$$[\lambda s'.at(p, x, s') \wedge raining(x, s')](s). \tag{2}$$

In QHL all these are expressible by different formulas without lambda abstraction. The fluent $\lambda s'.at(p, x, s') \wedge raining(x, s')$ is simply expressed in QHL by $at(p, x) \wedge raining(x)$. The formulas (1) and (2) are then expressed by $@_s(at(p, x) \wedge raining(x))$, an almost literal translation of the statement in natural language. Finally the original

[3] The holds operator plays an important role in a number of knowledge representation formalisms, for instance in Allen's work on events and intervals [1] and in Kowalski's event calculus [11].

formulation is expressed by distributing $@_s$ over the conjunction as in $@_s at(p,x) \wedge @_s raining(x)$.

Theories and definitions. There is a second reason why McCarthy and Hayes want explicit reference to situations. To express laws of nature, definitions or other information which is supposed to be true in all situations, you have to universally quantify over situations. They give the example of a kind of transitivity for the predicate *in(x,y,s)* which expresses that x is in the location in situation s:

$$\forall x \forall y \forall z \forall s.(in(x,y,s) \wedge in(y,z,s) \rightarrow in(x,z,s)) \tag{3}$$
$$\forall x \forall y \forall z \forall .(in(x,y) \wedge in(y,z) \rightarrow in(x,z)). \tag{4}$$

In the second statement the situation argument is suppressed and $\forall.$ is meant to implicitly quantify over all situations. In modal terminology $\forall.$ functions as a *universal modality*. In description logic a special status is given to statements which are supposed to be true in all situations. They are placed in, what is called, the *T–Box* (for Theory Box). This is the natural place to collect definitions and other laws which hold universally. We adopt this T–Box machinery and express (3) and (4) simply by putting the *QHL* sentence (5) in the T–Box.

$$\forall x \forall y \forall z(in(x,y) \wedge in(y,z) \rightarrow in(x,z)) \tag{5}$$

Note that this is almost literally the formulation (4) which is preferred in [14], except that the unappealing empty quantifier is replaced by the T–Box.

Prior's temporal operators. In section 2 of [14], Prior's temporal operator F is introduced in the situation calculus. Here it becomes clear that the used formalism is not suited: only with explicit λ–abstraction can one make a simple causality assertion. $\mathsf{F}(\pi, s)$ means that "the situation s will be followed (after an unspecified time) by a situation that satisfies the fluent π". To describe the temporal aspect of situations, McCarthy and Hayes postulate a function *time* from the set of situations to a set of time–points. The last set comes with the usual (linear) *earlier than* ordering.

Now (6) is the formalization of the assertion that "if a person is out in the rain, he will get wet".

$$\forall x \forall p \forall s[raining(x,s) \wedge at(p,x,s) \wedge outside(p,s) \rightarrow \mathsf{F}(\lambda s'.wet(p,s'),s)]. \tag{6}$$

This is also too much for McCarthy and Hayes and they quickly suppress explicit mention of situations, yielding

$$\forall x \forall p \forall .[raining(x) \wedge at(p,x) \wedge outside(p) \rightarrow \mathsf{F}(wet(p))]. \tag{7}$$

If we delete the empty quantifier $\forall.$ in (7) and put the result in the T–Box, we get the formalization in temporal *QHL*.

In temporal *QHL*, Prior's temporal operators F and P are added to the language: whenever ϕ is a formula, also $\mathsf{F}\phi$ and $\mathsf{P}\phi$ are formulas. Their meaning is evaluated locally in a situation: $\mathsf{F}\phi$ is true in a situation s if there exists a situation s' such that $time(s) < time(s')$ and ϕ is true at s'. The meaning of $\mathsf{P}\phi$ is defined similarly but with s' *before* s. Thus $\mathsf{F}\phi$ is true in a situation s if there exists a situation *in the future* of s at which ϕ is true. $\mathsf{P}\phi$ expresses the same thing, but with respect to the *past*.

Actions. The largest change in the language comes from our treatment of actions as compared to that in [14]. (A related approach is taken by Levesque, Pirri and Reiter [12], cf. also Reiter's book [18]). We treat actions as in dynamic logic [10] and introduce a modality for every action. McCarthy and Hayes [14] deal with actions through the situational fluent $\mathit{result}(p, \sigma, s)$. In this, p is a person, σ an action and s a situation. The value of $\mathit{result}(p, \sigma, s)$ is the situation that results when p carries out σ, starting in s. If the action does not terminate $\mathit{result}(p, \sigma, s)$ is considered undefined.

Note that $\mathit{result}(p, \sigma, s)$ is a function with the set of situations as its range. Using functions one can only handle deterministic actions. Another drawback of this representation is the use of partial functions. It is unclear what truth value a formula should receive when some of its arguments are undefined. Reiter [18] has similar problems which lead to the introduction of "ghost situations." Dynamic logic offers a solution for these problems, but pays the price that explicit reference to situations is not possible in the language. As we will see, when this is needed it can be elegantly done in hybrid logic. To simplify matters, we just consider actions and let the actor be implicit. So assume there is a set ACT of primitive actions. Then whenever ϕ is a formula and $\alpha \in \mathsf{ACT}$ is an action, also $\langle\alpha\rangle\phi$ and $[\alpha]\phi$ are formulas. $\langle\alpha\rangle\phi$ is true in a situation s if there exists a situation s' which is the result of carrying out α in s and ϕ is true in s'. $[\alpha]\phi$ is defined dually, so that ϕ needs to be true in *all* situations s' which result from carrying out α in s. Thus if α is a deterministic action $@_s[\alpha]\phi$ expresses that ϕ is true in the situation $\mathit{result}(\alpha, s)$.

McCarthy and Hayes use *result* to express certain laws of ability of the form $@_s\phi \rightarrow @_{s'}\psi$ with $s' = \mathit{result}(\sigma, s)$, expressing that if ϕ holds at s, then ψ is true in the situation which is the result of carrying out σ in s. With action modalities one can make more fine–grained distinctions. $@_s\phi \rightarrow \langle\alpha\rangle\top$ expresses that α *can* be carried out in situation s if ϕ holds there. $@_s\phi \rightarrow [\alpha]\psi$ expresses that *if* α is carried out in s under the assumption of ϕ, then ψ is true in every resulting situation (though there need not exist one). Here are two more examples of properties which cannot be expressed in situation calculus (or for that matter, in dynamic logic), but can in the hybrid formalism: $@_s\langle\alpha\rangle\top$ expresses that it is possible to carry out action α successfully in situation s; $@_s[\alpha]\mathsf{P}s$ expresses that the situation which results after carrying out action α in situation s is later in time than s. In plain words this formula expresses that it takes time to perform α. The combination of actions into strategies is immediate in this approach. Whenever ϕ is a formula and $\alpha_1, \ldots, \alpha_n \in \mathsf{ACT}$ are actions, also $\langle\alpha_1\rangle \cdots \langle\alpha_n\rangle\phi$ and $[\alpha_1] \cdots [\alpha_n]\phi$ are formulas.

4 Discussion and Conclusions

The seminal paper that McCarthy and Hayes [14] published in 1969 marks a watershed in artificial intelligence. Its importance can simply not be underestimated: apart from introducing the situation calculus as one of the main representation languages in artificial intelligence, the paper is most famous for singling out a number of fundamental problems that did set artificial intelligence's research agenda for years to come. Amongst its most important contributions are its role in the identification of the monotonicity of classical logic as a fundamental problem for intelligent robots; and perhaps it is most famous for introducing the frame problem (an area of unsurpassed activity in artificial intelligence). Both these fundamental problems resulted in important research traditions (see [7] for an overview of the field of non-monotonic reasoning, and see [19] for a survey of the frame problem). Nowadays, the ideas of [14] seem to have reached their ultimate success—they are part of the common knowledge and taken for granted by most researchers. Nevertheless, we feel that there are more than historical reasons for re-appraising [14].

A less frequently discussed contribution of the original paper is that it highlighted the connection between the fields of AI and philosophical logic (especially modal logics of belief, knowledge, and tense). This is even more extraordinary considering that the formulation in terms of Kripke semantics of these modal logics were recent developments in the 60s, and at that time part of a rather peripheral area in logic, plagued by deep philosophical problems. However, also modal logic progressed since the 60s and broadened its subject matter. As an illustration, the recent monograph [4] starts with stating that "modal languages are simple yet expressive languages for talking about relational structures". It is this view, of modal logic as a multi–purpose knowledge representation language, which holds the promise to shed new light on some of the fundamental problems of knowledge representation. Arthur Prior held this view already, now it is being fully developed in the fields of description logic [8] and hybrid logic [3].

The main contribution of this paper is a new quantified hybrid logic with temporal operators and action modalities, tailor-made for expressing the fluents of situation calculus. We have shown that in this quantified hybrid logic, situation calculus formulas can be expressed more natural and concise than the original formulations. Moreover, it comes with additional operators such as a downarrow binder that may enhance its expressive power beyond the original situation calculus. More generally speaking, the aim of this paper was to revive the connection between situation calculus and modal logic. This aim can perhaps best be viewed as an effort to bring back together two research traditions that have worked independently for many years. This may also help to highlight some of the common interests of knowledge representation and modal logic. We can only hope that this inspires further collaboration, and fruitful exchange of ideas between the two communities.

References

1. J. Allen. Maintaining knowledge about temporal intervals. *Artificial Intelligence*, 26:832–843, 1983.
2. C. Areces, P. Blackburn, and M. Marx. Hybrid logics. Characterization, interpolation and complexity. *Journal of Symbolic Logic*, 2001. In print.
3. P. Blackburn. Representation, reasoning, and relational structures: a hybrid logic manifesto. *Logic Journal of the IGPL*, 8(3):339–365, 2000.
4. P. Blackburn, M. de Rijke, and Y. Venema. *Modal Logic*, volume 53 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, Cambridge UK, 2001.
5. A. Borgida. Description logics in data management. *IEEE Transactions on Knowlede and Data Engineering*, 7:671–682, 1995.
6. R. Brachman and J. Schmolze. An overview of the KL–ONE knowledge representation system. *Cognitive Science*, 9(2):171–216, 1985.
7. G. Brewka, J. Dix, and K. Konolige. *Nonmonotonic Reasoning: an overview*. Number 73 in CSLI Lecture Notes. CSLI Publications, Stanford CA, 1997.
8. D. Calvanese, G. De Giacomo, D. Nardi, and M. Lenzerini. Reasoning in expressive description logics. In A. Robinson and A. Voronkov, editors, *Handbook of Automated Reasoning*. Elsevier Science Publishers, 1999.

9. M. Fitting. *Proof Methods for Modal and Intuitionistic Logics*. Number 169 in Synthese Library. Reidel, Dordrecht, 1983.
10. D. Harel, D. Kozen, and J. Tiuryn. *Dynamic Logic*. MIT Press, 2000.
11. R. A. Kowalski and M. J. Sergot. A logic based calculus of events. *New Generation Computing*, 4:67–95, 1986.
12. H. J. Levesque, F. Pirri, and R. Reiter. Foundations for a calculus of situations. *Electronic Transactions on Artificial Intelligence*, 2:159–178, 1998.
13. J. McCarthy. Situations, actions, and causal laws. Stanford Artificial Intelligence Project Memo 2, Stanford University, 1963.
14. J. McCarthy and P. J. Hayes. Some philosophical problems from the standpoint of artificial intelligence. In B. Meltzer and D. Michie, editors, *Machine Intelligence 4*, pages 463–502. Edinburgh University Press, 1969.
15. S. Passy and T. Tinchev. An essay in combinatory dynamic logic. *Information and Computation*, 93:263–332, 1991.
16. V. Pratt. Models of program logics. In *Proceedings of the 20th IEEE symposium on Foundations of Computer Science*, pages 115–122, 1979.
17. A. N. Prior. *Past, Present and Future*. Oxford University Press, 1967.
18. R. Reiter. Knowledge in action: Logical foundations for describing and implementing dynamical systems. Draft version of a book on situation calculus, 1996–2000.
19. M. Shanahan. *Solving the Frame Problem: A Mathematical Investigation of the Common Sense Law of Intertia*. The MIT Press, Cambridge MA, 1997.

Appendix: Formal Definition of Quantified Hybrid Logic

The language of quantified hybrid logic QHL has a set NOM of nominals, a set ACT of action statements, a set FVAR of first order variables, a set CON of first order constants, and predicates of any (including nullary) arity. The *terms* of the language are the constants from CON plus the first order variables from FVAR. The *atomic formulas* are all symbols in NOM together with the usual first order atomic formulas generated from the predicate symbols and equality using the terms. *Complex formulas* are generated from these according to the rules

$$@_n\phi \mid \neg\phi \mid \phi\wedge\psi \mid \phi\vee\psi \mid \phi\rightarrow\psi \mid \exists x\phi \mid \forall x\phi \mid \mathsf{F}\phi \mid \mathsf{P}\phi \mid \langle\alpha\rangle\phi \mid [\alpha]\phi$$

Here $n \in \mathsf{NOM}$, $x \in \mathsf{FVAR}$, and $\alpha \in \mathsf{ACT}$. These formulas are interpreted in situation calculus models. Such a model is a structure $(S, time, T, <, \{R_\alpha\}_{\alpha\in\mathsf{ACT}}, I_{nom}, D, I_{con}, I_s)_{s\in S}$ such that

- ○ S is a set of situations
- ○ $time$ is a function from S to the set of time points T
- ○ $(T, <)$ is a linearly ordered flow of time
- ○ $\{R_\alpha\}_{\alpha\in\mathsf{ACT}}$ is a set of binary relations on S, one for each action $\alpha \in \mathsf{ACT}$.
- ○ I_{nom} is a function assigning members of S to nominals;
- ○ I_{con} is a function assigning elements of D to constants in CON;
- ○ for each $s \in S$, (D, I_s) is an ordinary first order model

The satisfation relation, when is a formula ϕ true in situation s in model $\mathfrak{M}$ under the (variable) assignment g, simply follows the recursive construction of the language. For a full description of QHL with temporal operators and action modalities, the reader is refered to the long version of the paper on URL
⟨`http://www.illc.uva.nl/~kamps/sitcalc/`⟩.

A Modified Semantics for LUPS

João Alexandre Leite[⋆]

CENTRIA - Centro de Inteligência Artificial
Universidade Nova de Lisboa
2829-516 Caparica
Portugal
jleite@di.fct.unl.pt

Abstract. Following the introduction of *Dynamic Logic Programming* in [1], the language of updates *LUPS* was introduced in [2]. Whereas *Dynamic Logic Programming* provides a meaning to sequences of logic programs, each of them representing a state of the world, *LUPS* allows the specification of such states and state transitions.
In this paper, we take a closer look at the language *LUPS* and identify one problem with its semantics and a possible, important, extension to its set of commands. We then propose an extension to the syntax of *LUPS* as well as a new semantics that solves the identified problem. We illustrate the changes by means of two examples.

1 Introduction and Motivation

In the past few years, research in the area of *Nonmonotonic Reasoning* has devoted some attention to the problem of dynamically adapting a knowledge base (KB) to correctly represent a world that changes, i.e. how to update a KB. For the case where the KB is a theory in classical propositional logic, adequate solutions have been proposed in [8] and [15]. In [12,13], these solutions were adapted to allow for the update of logic programs and deductive databases, following the so called *interpretation update* approach. This approach has been shown inadequate when applied to nonmonotonic theories, often leading to counterintuitive results as pointed out in [9]. Since then, several approaches for updating KBs represented by logic programs have been proposed [1,3,4,9,10,14,16]. Each of these approaches proposes a semantics for what the outcome of the update of a logic program by another such program ought to be or, more generally, what the meaning of a sequence of logic programs should be. For considerations and comparisons wrt. these approaches see [4].

In [2], the authors argue that besides assigning a meaning to a sequence of logic programs, one also needs a language to specify how such sequence of programs is to be constructed i.e., besides declaratively specifying the states of a KB, it is advantageous to also declaratively specify the state transitions. And these state transitions should be allowed to depend on the states themselves. To this purpose, the language of updates *LUPS* [2] was introduced. In the *LUPS*

⋆ Partially supported by PRAXIS XXI scholarship no. BD/13514/97

framework, the next state of the KB is produced according to a set of commands and the KB at the current state. Such *LUPS* commands allow the specification of statements such as: "*a certain rule should belong to the next state if some condition holds in the current state*", which would be represented by the command "**assert** *Rule* **when** *Condition*". Similar commands exist for the retraction of rules, for the specification of permanent assert commands, i.e. an assert command that is to be executed at every state after being issued, and for the cancellation of such persistent commands. For further motivation for the need of a language as *LUPS*, and some application examples, see [2]. Throughout the remainder of this paper we assume that the reader is familiar with *LUPS* and its *Dynamic Logic Programming* (*DLP*) based semantics. In Appendix we provide an overview of *DLP* and *LUPS* with all the relevant intuitions and definitions.

1.1 Motivation

In *LUPS*, the authors introduced a class of commands whose immediate effect should only hold in the successor state and should not persist by inertia in subsequent states. This kind of non-inertial commands are indicated by the keyword **event** (e.g. "**assert event** *Rule* **when** *Condition*").

According to the intuitive reading above, if we want to update an initial KB, P_1, with two consecutive updates U_2 and U_3, such that U_2 only contains such non-inertial commands, and U_3 is empty, it seems reasonable to expect that after the second update, U_3, what holds true is exactly equal to what held true before the first update, i.e. what holds true at P_1. Unfortunately this is not the case in *LUPS*, as illustrated by the following example:

Example 1. Consider the simple case where $P_1 = \{a \leftarrow\}$, possibly obtained by a past update command such as **assert** $a \leftarrow$, and the following sequence of updates:

$$U_2 = \{\textbf{assert event } a \leftarrow\}$$
$$U_3 = \{\}$$

At state 3, i.e. after the update U_3, according to the semantics of *LUPS* we have $M_3 = \{\}$ as the only stable model, i.e. a is not a consequence of the knowledge base at state 3.

In this example, $M_3 = \{\}$ is the only stable model because the command **assert event** $a \leftarrow$ asserts the rule $a \leftarrow$ at state 2, but then causes the removal of all rules (past and present) of the form $a \leftarrow$, i.e. both the rule specified by U_2 and the rule of P_1 are removed. We argue that $M_3^{'} = \{a\}$ should be the only stable model at state 3, because the command **assert event** $a \leftarrow$ should not affect (remove at state 3) the rule $a \leftarrow$ that was previously asserted at state 1, i.e. the rule in P_1. Let us look at another example:

Example 2. Consider a slight modification in the previous example, such that the initial program is, now, $P_1^{*} = \{a \leftarrow \textit{not } \bot\}$ (where, as usual, $\bot$ is a reserved proposition with the property of being false in every stable model i.e, $not\ \bot$ belongs to every stable model). If the same update sequence is performed, after the update U_3 we have $M_3^* = \{a\}$ as the only stable model.

Updates are somewhat syntactical in nature (cf. [1]), but in this example, this syntactical difference in behaviour should not exist. We argue that both examples should have the same outcome, i.e. they should both have $M = \{a\}$ as the only stable model at state 3. To avoid such problem, we argue that an event command should be exerted on the single asserted (or retracted) instance of a rule and not on all other syntactically equal ones.

The main contribution of this paper is a proposal for a change in the semantics of *LUPS* to correct this undesirable behaviour.

The second contribution of this paper resides in the extension of the *LUPS* syntax with the introduction of a persistent retract command. In *LUPS*, besides the assertion and retraction commands, denoted by the keywords **assert** and **retract** respectively, that may only contribute to the state transition for which they were specified, there is a command, denoted by the keyword **always**, which can be seen as a permanent assert command, i.e. until it is cancelled, it will cause the assertion of a specific rule every time the specified condition holds. We argue that such persistent command should also exist for the retraction of rules. To illustrate our claim, let us consider the following example from [2]:

Example 3. Consider the following scenario: -once Republicans take over both Congress and the Presidency they establish a law stating that abortions are punishable by jail; -once Democrats take over both Congress and the Presidency they abolish such a law; -in the meantime, there are no changes in the law because always either the President or the Congress vetoes such changes. The specification in *LUPS*, as presented in [2], comprises the following persistent update commands[1]:

$$\textbf{always}\ jail(X) \leftarrow abortion(X)\ \textbf{when}\ repC, repP \tag{1}$$
$$\textbf{always}\ not\, jail(X) \leftarrow abortion(X)\ \textbf{when}\ not\, repC, not\, repP \tag{2}$$

The authors further state that in this example, alternatively, instead of the second command they could have used a retract command:

$$\textbf{retract}\ jail(X) \leftarrow abortion(X)\ \textbf{when}\ not\, repC, not\, repP \tag{3}$$

noting that, in this example, since there is no other rule implying *jail*, retracting the rule is safely equivalent to retracting its conclusion.

We argue that neither of the proposed solutions, to represent the abolition of the abortion law when the Democrats take over (commands (2) and (3)), properly represents the intuition stated in the scenario.

[1] Where the rules with variables simply stand, as usual, for all the ground rules that result from replacing the variables by all the ground terms in the language.

The argument against the use of command (2) is implicitly stated by the authors of [2] when they say that "*since there is no other rule implying jail, retracting the rule is safely equivalent to retracting its conclusion*". In fact, if there were other rules implying jail, which is fair to expect in a realistic scenario, such as for example a rule stating that assassinations are punishable by jail, then some undesirable effects would occur: -suppose such rule was represented by an update command, at the initial state, of the form:

$$\textbf{assert } jail(X) \leftarrow assassination(X) \tag{4}$$

If this command had been previously issued, then, after the Democrats take over both Congress and the Presidency, someone (*mary*) that both assassins someone and performs an abortion would *not* be punished by jail. This is so because the rule asserted by command (2) would reject, according to the DLP semantics, the previously asserted rule specified by command (4). The resulting knowledge base would have $not\, jail(mary)$ as a consequence.

The argument against the use of command (3) resides in the fact that, to effectively represent the intended meaning, this command would have to belong to every update i.e. it would have to be explicitly added to the specification of *every* state transition. Not only this does not add to the simplicity of the language but, also, the fact that the representation of the effect of Republicans taking over requires a single update command, and that the representation of the effect of Democrats taking over requires several update commands, one at each update, is somehow unintuitive. The introduction of the **always** command in [2] was justified with the need to avoid such consecutive repetitions of the **assert** command. We believe that such persistent command should also exist for the retraction of rules. A command that, until cancelled, retracts a specific rule from the KB every time the specified condition holds. In this paper we introduce such command.

Throughout, to differentiate the original *LUPS* language and the extended and modified version here proposed, we refer to the latter as $LUPS^*$.

The remainder is structured as follows: First, in Sect.2, we introduce the syntax of the extended language $LUPS^*$. In Sect.3 we propose a semantics for this extended language that corrects the above mentioned problem. In Sect.4 we illustrate with two examples. In Sect. 5 we compare both semantics by means of a desirable property. We then conclude in Sect.6.

2 $LUPS^*$ - Syntax

In this Section we formalize the language of $LUPS^*$. We will keep all the commands of *LUPS*, with a slight modification in the syntax of two of such commands, which will be explained below, and introduce the above mentioned persistent retract command and its corresponding cancellation command.

The language of $LUPS^*$ will contain the basic non-persistent commands for the assertion and retraction of rules, denoted by the keywords **assert** and

retract. They are of the form:

$$\textbf{assert } L \leftarrow L_1, \ldots, L_k \textbf{ when } L_{k+1}, \ldots, L_m \tag{5}$$

$$\textbf{retract } L \leftarrow L_1, \ldots, L_k \textbf{ when } L_{k+1}, \ldots, L_m \tag{6}$$

Both these commands can be made persistent, i.e. until cancelled they are added to all subsequent updates. We will identify such persistent commands by prefixing the non-persistent commands with the keyword **always**. They are thus of the form:

$$\textbf{always assert } L \leftarrow L_1, \ldots, L_k \textbf{ when } L_{k+1}, \ldots, L_m \tag{7}$$

$$\textbf{always retract } L \leftarrow L_1, \ldots, L_k \textbf{ when } L_{k+1}, \ldots, L_m \tag{8}$$

Note here the first modification in the syntax of a *LUPS* command. In *LUPS*, command (7) does not have the keyword **assert**, being identified by the keyword **always** alone. Since we are now introducing the persistent retraction command, this new notation becomes more symmetrical and therefore more intuitive.

All the previous commands are inertial. To obtain the corresponding non-inertial commands one simply adds the keyword **event** to obtain the following commands:

$$\textbf{assert event } L \leftarrow L_1, \ldots, L_k \textbf{ when } L_{k+1}, \ldots, L_m \tag{9}$$

$$\textbf{retract event } L \leftarrow L_1, \ldots, L_k \textbf{ when } L_{k+1}, \ldots, L_m \tag{10}$$

$$\textbf{always assert event } L \leftarrow L_1, \ldots, L_k \textbf{ when } L_{k+1}, \ldots, L_m \tag{11}$$

$$\textbf{always retract event } L \leftarrow L_1, \ldots, L_k \textbf{ when } L_{k+1}, \ldots, L_m \tag{12}$$

Just as in *LUPS* there is a cancellation command for the persistent command **always**, denoted by the keyword **cancel**, so there will be one in *LUPS** but now denoted by the keyword **cancel assert**, to simplify the introduction of a cancellation command for the persistent retraction command, which will be denoted by the keyword **cancel retract**. These two commands are:

$$\textbf{cancel assert } L \leftarrow L_1, \ldots, L_k \textbf{ when } L_{k+1}, \ldots, L_m \tag{13}$$

$$\textbf{cancel retract } L \leftarrow L_1, \ldots, L_k \textbf{ when } L_{k+1}, \ldots, L_m \tag{14}$$

The syntax of the update commands of *LUPS** is, formally:

Definition 1 (*LUPS** - Update Commands). *A* LUPS* update command*, or* command *for short, is a propositional expression of any of the forms*[2]

$$[\textbf{always}\ ]\,\textbf{assert}\,[\ \textbf{event}]\ \ R \textbf{ when } \phi \tag{15}$$

$$[\textbf{always}\ ]\,\textbf{retract}\,[\ \textbf{event}]\ \ R \textbf{ when } \phi \tag{16}$$

$$\textbf{cancel assert } R \textbf{ when } \phi \tag{17}$$

$$\textbf{cancel retract } R \textbf{ when } \phi \tag{18}$$

[2] Where [**a**] will be used for notational convenience denoting either the presence or absence of **a**.

where R is a rule of the form $L \leftarrow L_1, \ldots, L_k$ and ϕ is a (possibly empty) conjunction of literals, $L_{k+1}, \ldots, L_m$, where L and each L_i are literals. If ϕ is empty, we simply omit the **when** *keyword. We refer to those commands* without *(resp.* with*) the keyword* **event** *as* inertial *(resp.* non-inertial*) commands. We refer to those commands* with *(resp.* without*) the keyword* **always** *as* persistent *(resp.* non-persistent*) commands.*

The following table summarizes the correspondence between the syntax of $LUPS^*$ and that of $LUPS$:

$LUPS^*$		$LUPS$
assert [**event**]	$\leftrightarrow$	**assert** [**event**]
retract [**event**]	$\leftrightarrow$	**retract** [**event**]
always assert [**event**]	$\leftrightarrow$	**always** [**event**]
always retract [**event**]		non existing
cancel assert	$\leftrightarrow$	**cancel**
cancel retract		non existing

An update program in $LUPS^*$ is defined as follows:

Definition 2 ($LUPS^*$ - Update Program). *An update program in* LUPS* *is a finite sequence of updates, where an update is a set of commands of the form (15) to (18).*

3 $LUPS^*$ - Semantics

The semantics of $LUPS^*$ is defined by incrementally translating update programs into sequences of generalized logic programs and by considering the semantics of the DLP formed by them.

Let $\mathcal{U} = U_1 \otimes ... \otimes U_n$ be a $LUPS^*$ update program. At every state t we determine the corresponding DLP, $\Upsilon_t(\mathcal{U}) = \mathcal{P}_t$.

The translation of a $LUPS^*$ program into a dynamic program is similar to the one presented for $LUPS$. It is made by induction, starting from the empty program P_0 and, for each update U_t, given the already built dynamic program $P_0 \oplus \cdots \oplus P_{t-1}$, determining the resulting program $P_0 \oplus \cdots \oplus P_{t-1} \oplus P_t$. To cope with persistent update commands, as for the $LUPS$ semantics, we also consider a set containing all currently active persistent commands, although its definition must be extended to deal with the newly introduced persistent retraction commands. As in $LUPS$, the retraction of rules imposes its unique identification. Therefore, the language of the resulting dynamic logic program must be extended with a new propositional variable "$N(R)$" for every rule R appearing in the original $LUPS$ program. To properly handle non-inertial commands, we also need to uniquely associate those rules appearing in non-inertial commands with the states they belong to. To this end, the language of the resulting dynamic logic program must also be extended with a new propositional variable "$Ev(R,S)$" for every rule R appearing in a non-inertial command in the original $LUPS$ program, and every state S.

We now present the translation for *LUPS**:

Definition 3 (Translation into dynamic logic programs). *Let $\mathcal{U} = U_1 \otimes \cdots \otimes U_n$ be a* LUPS* *update program. The corresponding dynamic logic program $\Upsilon^*(\mathcal{U}) = \mathcal{P} = \mathcal{P}_n = P_0 \oplus \cdots \oplus P_n$ is obtained by the following inductive construction, using at each step t an auxiliary set of persistent commands PC_t:*

Base step: $P_0 = \{\}$ with $PC_0 = \{\}$.

Inductive step: Let $\Upsilon^*_{t-1}(\mathcal{U}) = \mathcal{P}_{t-1} = P_0 \oplus \cdots \oplus P_{t-1}$ with the set of persistent commands PC_{t-1} be the translation of $\mathcal{U}_{t-1} = U_1 \otimes \cdots \otimes U_{t-1}$. The translation of $\mathcal{U}_t = U_1 \otimes \cdots \otimes U_t$ is $\Upsilon^*_t(\mathcal{U}) = \mathcal{P}_t = P_0 \oplus \cdots \oplus P_{t-1} \oplus P_t$ with the set of persistent commands PC_t, where:

$$\begin{aligned}
PC_t = PC_{t-1} &\cup \{\textbf{assert } R \textbf{ when } \phi : \textbf{always assert } R \textbf{ when } \phi \in U_t\} \cup \\
&\cup \{\textbf{retract } R \textbf{ when } \phi : \textbf{always retract } R \textbf{ when } \phi \in U_t\} \cup \\
&\cup \{\textbf{assert event } R \textbf{ when } \phi : \textbf{always assert event } R \textbf{ when } \phi \in U_t\} \cup \\
&\cup \{\textbf{retract event } R \textbf{ when } \phi : \textbf{always retract event } R \textbf{ when } \phi \in U_t\} \cup \\
&- \{\textbf{assert } [\textbf{event}]\ R \textbf{ when } \phi : \textbf{cancel assert } R \textbf{ when } \psi \in U_t \wedge \\
&\qquad \wedge \textstyle\bigoplus \mathcal{P}_{t-1} \vDash \psi\} - \\
&- \{\textbf{assert } [\textbf{event}]\ R \textbf{ when } \phi : \textbf{always retract } [\textbf{event}]\ R \textbf{ when } \psi \in U_t \wedge \\
&\qquad \wedge \textstyle\bigoplus \mathcal{P}_{t-1} \vDash \psi\} - \\
&- \{\textbf{retract } [\textbf{event}]\ R \textbf{ when } \phi : \textbf{cancel retract } R \textbf{ when } \psi \in U_t \wedge \\
&\qquad \wedge \textstyle\bigoplus \mathcal{P}_{t-1} \vDash \psi\} - \\
&- \{\textbf{retract } [\textbf{event}]\ R \textbf{ when } \phi : \textbf{always assert } [\textbf{event}]\ R \textbf{ when } \psi \in U_t \wedge \\
&\qquad \wedge \textstyle\bigoplus \mathcal{P}_{t-1} \vDash \psi\}
\end{aligned}$$

$$NU_t = U_t \cup PC_t$$

$$\begin{aligned}
P_t = &\{not\, N(R) \leftarrow : \textbf{retract } R \textbf{ when } \phi \in NU_t \wedge \textstyle\bigoplus \mathcal{P}_{t-1} \vDash \phi\} \cup \\
\cup &\{N(R) \leftarrow; H(R) \leftarrow B(R), N(R) : \textbf{assert } R \textbf{ when } \phi \in NU_t \wedge \textstyle\bigoplus \mathcal{P}_{t-1} \vDash \phi\} \cup \\
\cup &\{H(R) \leftarrow B(R), Ev(R,t) : \textbf{assert event } R \textbf{ when } \phi \in NU_t \wedge \textstyle\bigoplus \mathcal{P}_{t-1} \vDash \phi\} \cup \\
\cup &\{not\, N(R) \leftarrow Ev(R,t) : \textbf{retract event } R \textbf{ when } \phi \in NU_t \wedge \textstyle\bigoplus \mathcal{P}_{t-1} \vDash \phi\} \cup \\
\cup &\{not\, Ev(R, t-1) \leftarrow; Ev(R,t) \leftarrow\}
\end{aligned}$$

Note that the body of every rule R must be extended either with the predicate $N(R)$ or with the predicate $Ev(R,t)$. The differences between this transformation and the original *LUPS* transformation are the following: - the modification in the definition of the set of active persistent commands, PC_t, to deal with the newly introduced persistent retraction command and the corresponding cancellation command; - the modification in the definition of the generalized logic program at state t, P_t, to properly deal with non-inertial commands. This is achieved by treating inertial and non-inertial rules in a separate manner: the former are dealt with as in LUPS while the latter are extended with the predicate $Ev(R,t)$ which is only made true for the duration of one state. This is achieved simply by including the rules $not\, Ev(R, t-1) \leftarrow$ and $Ev(R,t) \leftarrow$ in the generalized logic program of every state.

The semantics of *LUPS** is, as expected:

Definition 4 (*LUPS** Semantics). *Let $\mathcal{U} = U_1 \otimes ... \otimes U_n$ be a* LUPS* *program. A query* **holds** $L_1, \ldots, L_k$ **at** t? *is true in $\mathcal{U}$ iff $\bigoplus \Upsilon_t^* (\mathcal{U}) \vDash L_1, \ldots, L_k$, or, equivalently, iff $\bigoplus \mathcal{P}_t \vDash L_1, \ldots, L_k$. By $SM(\mathcal{U})$ we mean the set of all stable models of $\Upsilon^*(\mathcal{U})$ at state n, modulo the newly introduced literals $N(R)$ and $Ev(R, S)$.*

4 Illustrative Examples

Follows an example to illustrate the different behaviour between *LUPS* and *LUPS**:

Example 4. Consider a public building with several floors. Initially, the security policy of the building states that any person (P) is allowed into any of its floors (F) only if they have a special permit for that floor. This can be represented by the update U_1:

$$\begin{aligned} U_1 : &\ \textbf{assert}\ (allowed(P, F) \leftarrow permission(P, F)) \\ &\ \textbf{assert}\ (not\, allowed(P, F) \leftarrow not\, permission(P, F)) \end{aligned}$$

Later on, the administration decided to open up a public relations office, to be situated in the ground floor. They had to update the security policy which, from then on, would allow any person in the ground floor provided they have some form of identification. Let us suppose this happened in the second day (state), and is represented by the update U_2:

$$U_2 : \textbf{assert}\ (allowed(P, ground) \leftarrow id(P))$$

Further down the line (at the third day), the administration decided to, once every now and then, declare an open day when everyone, for the duration of one day, was allowed to visit the entire building, provided they have some form of Id. This is represented by the update U_3:

$$U_3 : \textbf{always assert event}\ \ (allowed(P, F) \leftarrow id(P))\ \ \textbf{when}\ open\_day$$

Suppose that both Mary and John have Ids and Mary has a permission for the second floor, represented by the facts $permission(mary, second)$, $id(john)$ and $id(mary)$, in the KB. At day five there is an open day represented by the update at state 4:

$$U_4 : \textbf{assert event}\ \ (open\_day \leftarrow)$$

According to the *LUPS** semantics, at state 4 Mary is allowed in the ground and second floors, and John is only allowed in the ground floor; at state 5 both Mary and John are allowed in all floors of the building; at state 6 and thereafter everything is back as it was at state 4, with Mary being allowed in the ground and second floors, and John only being allowed in the ground floor, as expected. If this problem was to be tackled with the semantics of *LUPS*, everything would be the same until (and including) state 5 but, at state 6, John and Mary (and anybody else) would no longer be allowed in the ground floor, which seems to be rather unintuitive. This is so because not only the rule asserted by the **always assert event** command in U_3 is removed, but so is the rule asserted by U_2, simply because it has the same syntax. □

We now show how Example 3 would be encoded in $LUPS^*$:

Example 5. Consider the scenario of Example 3. The specification in $LUPS^*$ would comprise the following persistent update commands:

$$\textbf{always assert } jail(X) \leftarrow abortion(X) \textbf{ when } repC, repP$$
$$\textbf{always retract } jail(X) \leftarrow abortion(X) \textbf{ when } not\, repC, not\, repP$$

If, as before, at the initial state we also have the following update command:

$$\textbf{assert } jail(X) \leftarrow assassination(X)$$

Then, after the Democrats take over both Congress and the Presidency, if Mary both assassins someone and performs an abortion, she will now be punished by jail, as intended. □

5 Comparison

In this section we compare $LUPS$ and $LUPS^*$ by means of a property that partially characterizes the desired behaviour of the semantics wrt. non-inertial commands.

Before we express this property, we start with the definition of *update equivalence* for update programs, according to which two update programs are *update equivalent* iff their semantics coincides after an arbitrary number of updates with arbitrary update programs. Formally:

Definition 5 (Update Equivalence). *Let $\mathcal{U}_1$ and $\mathcal{U}_2$ be two update programs. We say that $\mathcal{U}_1$ and $\mathcal{U}_2$ are* update equivalent*, denoted by $\mathcal{U}_1 \overset{\otimes}{\equiv} \mathcal{U}_2$, iff for every update program $\mathcal{Q}$, $\mathcal{U}_1 \otimes \mathcal{Q}$ is semantically equivalent to $\mathcal{U}_2 \otimes \mathcal{Q}$, i.e.:*

$$SM\,(\mathcal{U}_1 \otimes \mathcal{Q}) = SM\,(\mathcal{U}_2 \otimes \mathcal{Q})$$

where if $\mathcal{R} = R_{r_1} \otimes ... \otimes R_{r_n}$ and $\mathcal{S} = S_{s_1} \otimes ... \otimes S_{s_m}$ are two update programs, by $\mathcal{R} \otimes \mathcal{S}$ we mean the update program $R_{r_1} \otimes ... \otimes R_{r_n} \otimes S_{s_1} \otimes ... \otimes S_{s_m}$.

With this property, we can now come back to our main claim. In particular, we claim that if we have a sequence of updates such that only non-inertial commands are executed, the knowledge state before the execution of such sequence of commands and the knowledge state after the execution of all such commands should be *update equivalent.* This means that the long term effect (more than one state) of non-inertial commands should only reside in their interaction with inertial commands, be them persistent or not, i.e., rules asserted or retracted by non-inertial commands should only change the semantics at the next state to allow and/or prevent the execution of other commands at that state, which may be inertial and therefore change the semantics at subsequent states. In particular, if there is a sequence of updates without any inertial command to be executed, followed by a state transition specified by an empty update, then, after its execution the knowledge state should be *equivalent* to the initial one. In other words, we want the effect of non-inertial commands to be, in fact, non inertial. The referred empty update is necessary because the immediate effect of every non-inertial command holds for one state, i.e. we need an extra state transition for such effect to be removed.

This is formally defined as follows:

Definition 6 (Stability wrt non-inertial commands). *An update language is* stable wrt. non-inertial commands *iff for every update programs $\mathcal{U}_1$ and $\mathcal{U}_2$ such that $\mathcal{U}_1$ consists of updates with non-persistent commands only, and $\mathcal{U}_2$ consists of updates with non-persistent, non-inertial commands only,*

$$\mathcal{U}_1 \otimes U_\emptyset \overset{\otimes}{\equiv} \mathcal{U}_1 \otimes \mathcal{U}_2 \otimes U_\emptyset$$

where $U_\emptyset$ denotes an empty update.

The restriction imposed on the updates of $\mathcal{U}_1$ to only contain non-persistent commands, i.e. without the keyword **always**, is justified by the fact that if such persistent commands were present, they could be executed at subsequent states, together with the updates of $\mathcal{U}_2$, thus invalidating our goal to have only non-inertial commands being executed at the state transitions corresponding to the updates of $\mathcal{U}_2$. Recall that an inertial command is valid at all subsequent states, until cancelled. Note that to express our intuition it suffices to guarantee that at the state transitions corresponding to the updates of $\mathcal{U}_2$ only non-inertial commands are executed. In fact, this would also be achieved by allowing $\mathcal{U}_1$ to contain persistent non-inertial commands so as long as we ensure that at the state transition corresponding to $U_\emptyset$ no commands are executed, but we will keep to this simpler formulation.

Proposition 1. LUPS *is not stable wrt. non-inertial commands.*

Example 1 above shows that *LUPS* is not stable wrt. non-inertial commands. It is worth noting that the recently proposed language of updates *EPI* [5], being based on the semantics of *LUPS*, is also not stable wrt. non-inertial commands. The same example also applies to *EPI*.

Theorem 1. LUPS* *is stable wrt. non-inertial commands.*

6 Conclusion

In this paper we have drawn the attention of the reader to an intuitively incorrect behaviour of the semantics of the language of updates *LUPS*, when dealing with non-inertial commands. Hoping to have convinced the reader that such behaviour is in fact incorrect, we have then proposed a modification to the semantics of *LUPS* to correct such problem. Both semantics, the original and the modified, were compared by means of a desirable property that only holds in the latter.

We have also suggested the need for a new persistent retraction command, symmetrical to the persistent assertion command, and extended the syntax and adapted the semantics accordingly.

Each contribution was illustrated by means of an example.

References

1. J. J. Alferes, J. A. Leite, L. M. Pereira, H. Przymusinska, and T. Przymusinski. Dynamic updates of non-monotonic knowledge bases. *Journal of Logic Programming*, 45(1-3):43–70, 2000. Abstract titled *Dynamic Logic Programming* appeared in Procs. of KR-98.
2. J. J. Alferes, L. M. Pereira, H. Przymusinska, and T. Przymusinski. LUPS : A language for updating logic programs. In *Procs. of LPNMR-99*, LNAI-1730. Springer, 1999.
3. F. Buccafurri, W. Faber, and N. Leone. Disjunctive logic programs with inheritance. In *Procs. of ICLP-99*. MIT Press, 1999.
4. T. Eiter, M. Fink, G. Sabbatini, and H. Tompits. Considerations on updates of logic programs. In *Procs. of JELIA-00*, LNAI-1919. Springer, 2000.
5. T. Eiter, M. Fink, G. Sabbatini, and H. Tompits. Specifying update policies for nonmonotonic knowledge bases. In *Procs. of DGNMR-01*, 2001. To appear in Procs. of IJCAI-01.
6. M. Gelfond and V. Lifschitz. The stable semantics for logic programs. In *Procs. of ICLP-88*. MIT Press, 1988.
7. K. Inoue and C. Sakama. Negation as failure in the head. *Journal of Logic Programming*, 35:39–78, 1998.
8. H. Katsuno and A. Mendelzon. On the difference between updating a knowledge base and revising it. In *Procs. of KR-91*. Morgan Kaufmann, 1991.
9. J. A. Leite and L. M. Pereira. Generalizing updates: From models to programs. In *Procs of. LPKR-97*, LNAI-1471. Springer, 1997.
10. J. A. Leite and L. M. Pereira. Iterated logic program updates. In *Procs. of JICSLP-98*. MIT Press, 1998.
11. V. Lifschitz and T. Woo. Answer sets in general non-monotonic reasoning (preliminary report). In *Procs. of KR-92*. Morgan-Kaufmann, 1992.
12. V. W. Marek and M. Truszczyński. Revision specifications by means of programs. In *Procs. of JELIA-94*, LNAI-838. Springer, 1994.
13. T. C. Przymusinski and H. Turner. Update by means of inference rules. *Journal of Logic Programming*, 30(2):125–143, 1997.
14. C. Sakama and K. Inoue. Updating extended logic programs through abduction. In *Procs. of LPNMR-99*. Springer, 1999.
15. M. Winslett. Reasoning about action using a possible models approach. In *Procs. of NCAI-88*. AAAI Press, 1988.
16. Y. Zhang and N. Foo. Updating logic programs. In *Procs. of ECAI'98*. Morgan Kaufmann, 1998.

A Background

In this Appendix we provide some background on Generalized Logic Programs, Dynamic Logic Programming and *LUPS*.

Generalized Logic Programs Here we recapitulate the syntax and stable semantics of generalized logic programs[3] [1].

By a *generalized logic program* P in a language $\mathcal{L}$ we mean a finite or infinite set of propositional clauses of the form $L_0 \leftarrow L_1, \ldots, L_n$ where each L_i is a literal (i.e. an atom A or the default negation of an atom $not\,A$). If r is a clause (or rule), by $H(r)$ we mean L_0, and by $B(r)$ we mean $L_1, \ldots, L_n$. If $H(r) = A$ (resp. $H(r) = not\,A$) then $not\,H(r) = not\,A$ (resp. $not\,H(r) = A$). By a (2-valued) *interpretation* M of $\mathcal{L}$ we mean any set of literals from $\mathcal{L}$ that satisfies the condition that for any A, *precisely one* of the literals A or $not\,A$ belongs to M. Given an interpretation M we define $M^+ = \{A : A \text{ is an atom}, A \in M\}$ and $M^- = \{not\,A : A \text{ is an atom}, \; not\,A \in M\}$. Wherever convenient we omit the default (negative) atoms when describing interpretations and models. Also, rules with variables stand for the set of their ground instances. We say that a (2-valued) interpretation M of $\mathcal{L}$ is a stable model of a generalized logic program P if $\rho(M) = least\,(\rho(P) \cup \rho(M^-))$, where $\rho(.)$ univocally renames every default literal $not\,A$ in a program or model into new atoms, say not_A. In the remaining, we refer to a GLP simply as a logic program (or LP).

Dynamic Logic Programming Next we recall the semantics of *dynamic logic programming* [1]. A dynamic logic program $\mathcal{P} = \{P_s : s \in S\} = P_0 \oplus ... \oplus P_n \oplus ...$, is a finite or infinite sequence of LPs, indexed by the finite or infinite set $S = \{1, 2, \ldots, n, \ldots\}$. Such sequence may be viewed as the outcome of updating P_0 with P_1, ..., updating it with P_n,... As we will see, in *LUPS* each P_i is determined by the i^{th} state transition. The role of dynamic logic programming is to ensure that these newly added rules are in force, and that previous rules are still valid (by inertia) for as long as they do not conflict with more recent ones. The notion of dynamic logic program at state s, denoted by $\bigoplus_s \mathcal{P} = P_0 \oplus ... \oplus P_s$, characterizes the meaning of the dynamic logic program when queried at state s, by means of its stable models, defined as follows:

Definition 7 (Stable Models of DLP). *Let* $\mathcal{P}$ =$\{P_s : s \in S\}$ *be a* dynamic logic program, *let* $s \in S$. *An interpretation* M_s *is a* stable model of $\mathcal{P}$ at state s *iff*

$$M_s = least\,([\mathcal{P}_s - Reject(s, M_s)] \cup Default\,(\mathcal{P}_s, M_s))$$

[3] The class of GLPs (i.e. logic programs that allow default negation in the premisses and heads of rules) can be viewed as a special case of yet broader classes of programs, introduced earlier in [7] and in [11], and, for the special case of normal programs, their semantics coincides with the stable models semantics [6].

where

$$\mathcal{P}_s = \bigcup_{i \le s} P_i$$
$$Reject(s, M_s) = \{r \in P_i : \exists r' \in P_j, i < j \le s, H(r) = not\, H(r') \wedge M_s \vDash B(r')\}$$
$$Default\,(\mathcal{P}_s, M_s) = \{not\, A \mid \nexists r \in \mathcal{P}_s : (H(r) = A) \wedge M_s \vDash B(r)\}$$

If some literal or conjunction of literals ϕ holds in all stable models of $\mathcal{P}$ at state s, we write $\bigoplus_s \mathcal{P} \vDash \phi$. If s is the largest element of S we simply write $\bigoplus \mathcal{P} \vDash \phi$.

LUPS Here we recall the language of updates *LUPS* closely following its original formulation in [2]. The object language of *LUPS* is that of generalized logic programs. A sentence U in *LUPS* is a set of simultaneous update commands (or actions) that, given a pre-existing sequence of logic programs $P_0 \oplus \cdots \oplus P_n$ (i.e. a dynamic logic program), whose semantics corresponds to our knowledge at a given state, produces a sequence with one more program, $P_0 \oplus \cdots \oplus P_n \oplus P_{n+1}$, corresponding to the knowledge that results from the previous sequence after performing all the simultaneous commands. A program in *LUPS* is a sequence of such sentences.

Given a program in *LUPS*, its semantics is defined by means of a dynamic logic program generated by the sequence of commands.

In this update framework, knowledge evolves from one knowledge state to another as a result of update commands stated in the object language. Without loss of generality it is assumed that the initial knowledge state is empty. Given the *current knowledge state*, its *successor knowledge state* is produced as a result of the occurrence of a set U of simultaneous *updates*. The knowledge state obtained by performing the sequence of updates $U_1, U_2, \ldots, U_n$ is denoted by $U_1 \otimes U_2 \otimes \cdots \otimes U_n$. So defined sequences of updates will be called *update programs*. In other words, an update program is a finite sequence $\mathcal{U} = \{U_s : s \in S\}$ of updates indexed by the set $S = \{1, 2, \ldots, n\}$. Each update is a set of update commands. Update commands (defined below) specify *assertions* or *retractions* to the current knowledge state. By the current knowledge state we mean the one resulting from the last update performed.

Knowledge can be queried at any state $t \le n$, where n is the index of the current knowledge state. A query will be denoted by:

$$\textbf{holds}\ L_1, \ldots, L_k\ \textbf{at}\ t?$$

and is true iff the conjunction of its literals holds at the state obtained after the t^{th} update. If $t = n$, the state reference "**at** t" is skipped.

Update commands specify assertions or retractions to the current knowledge state. In *LUPS* a simple assertion is represented by the command:

$$\textbf{assert}\ L \leftarrow L_1, \ldots, L_k\ \textbf{when}\ L_{k+1}, \ldots, L_m \tag{19}$$

Its meaning is that if $L_{k+1}, \ldots, L_m$ is true in the current state, then the rule $L \leftarrow L_1, \ldots, L_k$ is added to its successor state, and persists by inertia, until possibly retracted or overridden by some future update command.

In order to represent rules and facts that do not persist by inertia, i.e. that are one-state events, LUPS includes the modified form of assertion:

$$\textbf{assert event } L \leftarrow L_1, \ldots, L_k \textbf{ when } L_{k+1}, \ldots, L_m \tag{20}$$

The retraction of rules is performed with the two update commands:

$$\textbf{retract } L \leftarrow L_1, \ldots, L_k \textbf{ when } L_{k+1}, \ldots, L_m \tag{21}$$

$$\textbf{retract event } L \leftarrow L_1, \ldots, L_k \textbf{ when } L_{k+1}, \ldots, L_m \tag{22}$$

Its meaning is that, subject to precondition $L_{k+1}, \ldots, L_m$ (verified at the current state) rule $L \leftarrow L_1, \ldots, L_k$ is either retracted from its successor state onwards, or just temporarily retracted in the successor state (if governed by **event**).

Normally assertions represent newly incoming information. Although its effects remain by inertia (until contravened or retracted), the assert command itself does not persist. However, some update commands may desirably persist in the successive consecutive updates. This is especially the case of laws which, subject to some preconditions, are always valid, or of rules describing the effects of an action. In the former case, the update command must be added to all sets of updates, to guarantee that the rule remains indeed valid. In the latter case, the specification of the effects must be added to all sets of updates, to guarantee that, when the action takes place, its effects are enforced.

To specify such persistent update commands, *LUPS* introduces the following commands:

$$\textbf{always } L \leftarrow L_1, \ldots, L_k \textbf{ when } L_{k+1}, \ldots, L_m \tag{23}$$

$$\textbf{always event } L \leftarrow L_1, \ldots, L_k \textbf{ when } L_{k+1}, \ldots, L_m \tag{24}$$

$$\textbf{cancel } L \leftarrow L_1, \ldots, L_k \textbf{ when } L_{k+1}, \ldots, L_m \tag{25}$$

The first two statements mean that, in addition to any new set of arriving update commands, the persistent update command keep executing with them too. In the first case without, and in the second one with the **event** keyword. The third statement cancels execution of this persistent update, once the conditions for cancellation are met.

Definition 8 (LUPS). *An update program in LUPS is a finite sequence of updates, where an update is a set of commands of the form (19) to (25).*

The semantics of *LUPS* is defined by incrementally translating update programs into sequences of generalized logic programs and by considering the semantics of the *DLP* formed by them.

Let $\mathcal{U} = U_1 \otimes ... \otimes U_n$ be a *LUPS* programs. At every state t the corresponding *DLP*, $\Upsilon_t(\mathcal{U}) = \mathcal{P}_t$, is determined.

The translation of a *LUPS* program into a dynamic program is made by induction, starting from the empty program P_0, and for each update U_t, given the already built dynamic program $P_0 \oplus \cdots \oplus P_{t-1}$, determining the resulting program $P_0 \oplus \cdots \oplus P_{t-1} \oplus P_t$. To cope with persistent update commands, associated with every dynamic program in the inductive construction, a set containing all currently active persistent commands is considered, i.e. all those commands that were not cancelled until that point in the construction, from the time they were introduced. To be able to retract rules, a unique identification of each such rule is needed. This is achieved by augmenting the language of the resulting dynamic program with a new propositional variable "$N(R)$" for every rule R appearing in the original *LUPS* program.

Definition 9 (Translation into dynamic logic programs). *Let $\mathcal{U} = U_1 \otimes \cdots \otimes U_n$ be an update program. The corresponding dynamic logic program $\Upsilon(\mathcal{U}) = \mathcal{P} = P_0 \oplus \cdots \oplus P_n$ is obtained by the following inductive construction, using at each step t an auxiliary set of persistent commands PC_t:*

Base step: $P_0 = \{\}$ with $PC_0 = \{\}$.

Inductive step: Let $\Upsilon_{t-1}(\mathcal{U}) = \mathcal{P}_{t-1} = P_0 \oplus \cdots \oplus P_{t-1}$ with the set of persistent commands PC_{t-1} be the translation of $\mathcal{U}_{t-1} = U_1 \otimes \cdots \otimes U_{t-1}$. The translation of $\mathcal{U}_t = U_1 \otimes \cdots \otimes U_t$ is $\Upsilon_t(\mathcal{U}) = \mathcal{P}_t = P_0 \oplus \cdots \oplus P_{t-1} \oplus P_t$ with the set of persistent commands PC_t, where:

$$\begin{aligned}
&PC_t = PC_{t-1} \cup \{\textbf{assert } R \textbf{ when } \phi : \textbf{always } R \textbf{ when } \phi \in U_t\} \cup \\
&\cup \{\textbf{assert event } R \textbf{ when } \phi : \textbf{always event } R \textbf{ when } \phi \in U_t\} \cup \\
&- \{\textbf{assert } [\textbf{event}]\ R \textbf{ when } \phi : \textbf{cancel } R \textbf{ when } \psi \in U_t \wedge \bigoplus \mathcal{P}_{t-1} \vDash \psi\} - \\
&- \{\textbf{assert } [\textbf{event}]\ R \textbf{ when } \phi : \textbf{retract } R \textbf{ when } \psi \in U_t \wedge \bigoplus \mathcal{P}_{t-1} \vDash \psi\} -
\end{aligned}$$

$$NU_t = U_t \cup PC_t$$

$$\begin{aligned}
&P_t = \{N(R) \leftarrow; H(R) \leftarrow B(R), N(R) : \textbf{assert } [\textbf{event}]\ R \textbf{ when } \phi \in NU_t \wedge \\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \wedge \bigoplus \mathcal{P}_{t-1} \vDash \phi\} \cup \\
&\cup \{not\, N(R) \leftarrow: \textbf{retract } [\textbf{event}]\ R \textbf{ when } \phi \in NU_t \wedge \bigoplus \mathcal{P}_{t-1} \vDash \phi\} \cup \\
&\cup \{not\, N(R) \leftarrow: \textbf{assert event } R \textbf{ when } \phi \in NU_{t-1} \wedge \bigoplus \mathcal{P}_{t-2} \vDash \phi\} \cup \\
&\cup \{N(R) \leftarrow: \textbf{retract event } R \textbf{ when } \phi \in NU_{t-1} \wedge \bigoplus \mathcal{P}_{t-2} \vDash \phi, N(R)\}
\end{aligned}$$

Definition 10 (LUPS Semantics). *Let $\mathcal{U}$ be an update program. A query*

$$\textbf{holds } L_1, \ldots, L_n \textbf{ at } t$$

is true in $\mathcal{U}$ iff $\bigoplus_t \Upsilon(\mathcal{U}) \vDash L_1, \ldots, L_n$.

On the Use of Multi-dimensional Dynamic Logic Programming to Represent Societal Agents' Viewpoints

João Alexandre Leite, José Júlio Alferes, and Luís Moniz Pereira

Centro de Inteligência Artificial - CENTRIA
Universidade Nova de Lisboa, 2829-516 Caparica, Portugal
{jleite|jja|lmp}@di.fct.unl.pt

Abstract. This paper explores the applicability of the new paradigm of *Multi-dimensional Dynamic Logic Programming* to represent an agent's view of the combination of societal knowledge dynamics. The representation of a dynamic society of agents is the core of $\mathcal{MINERVA}$ [11], an agent architecture and system designed with the intention of providing a common agent framework based on the unique strengths of *Logic Programming*, hat allows the combination of several non-monotonic knowledge representation and reasoning mechanisms developed in recent years.

1 Introduction

Over recent years, the notion of agency has claimed a major role in defining the trends of modern research. Influencing a broad spectrum of disciplines such as Sociology, Psychology, among others, the agent paradigm virtually invaded every sub-field of Computer Science [3,8,16]. Although commonly implemented by means of imperative languages, mainly for reasons of efficiency, the agent concept has recently increased its influence in the research and development of computational logic based systems. Since efficiency is not always the crucial issue, but clear specification and correctness is, *Logic Programming* and *Non-monotonic Reasoning* have been brought back into the spotlight.

The *Logic Programming* paradigm provides a well-defined, general, integrative, encompassing, and rigorous framework for systematically studying computation, be it syntax, semantics, procedures, or attending implementations, environments, tools, and standards. *LP* approaches problems, and provides solutions, at a sufficient level of abstraction so that they generalize from problem domain to problem domain. This is afforded by the nature of its very foundation in logic, both in substance and method, and constitutes one of its major assets. To this accrues the recent significant improvements in the efficiency of *Logic Programming* implementations for *Non-monotonic Reasoning* (e.g. [14,18]). Besides allowing for a unified declarative and procedural semantics, eliminating the traditional wide gap between theory and practice, the use of several and quite powerful results in the field of non-monotonic extensions to *Logic Programming*

(LP), such as belief revision, inductive learning, argumentation, preferences, abduction, etc.[16] can represent an important composite added value to the design of rational agents.

Until recently, *Logic Programming* could be seen as a good representation language for static knowledge. If we are to move to a more open and dynamic environment, typical of the agency paradigm, we need to consider ways of representing and integrating knowledge from different sources which may evolve in time. Moreover, an agent not only comprises knowledge about states, but also some form of knowledge about the transitions between states. This knowledge about state transitions can represent the agent's knowledge about the environment's evolution, as well as its own behaviour and evolution. Since logic programs describe knowledge states, it's only fit that logic programs describe transitions of knowledge states as well. It is natural to associate with each state a set of transition rules to obtain the next state. Recent developments have opened *Logic Programming* to these otherwise unreachable dynamic worlds [1,4,6,17,19].

In [1], the authors, with others, introduced *Dynamic Logic Programming.* There, they studied and defined the declarative and operational semantics of sequences of logic programs (or dynamic logic programs). Each program in the sequence contains knowledge about some given state, where different states may, for example, represent different time periods or different sets of priorities. The introduction of *Dynamic Logic Programming* has extended Logic Programming, making possible for a logic program to undergo a sequence of modifications, opening up the possibility of incremental design and evolution of logic programs, therefore significantly facilitating *modularization* of logic programming and, thus, modularization of non-monotonic reasoning as a whole.

In [2], the authors, with others, introduced the language *LUPS* – "Language for dynamic updates" – designed for specifying changes to logic programs. Given an initial knowledge base (as a logic program) LUPS provides a way for sequentially updating it, unifying states and state transitions into a single declarative logic based framework.

Even though the main motivation behind the introduction of *Dynamic Logic Programming* was to represent the evolution of knowledge in time, the relationship between the different states can encode other aspects of a system, as explored in [1,9,5,15,12]. Although *Dynamic Logic Programming* can represent several states in one evolving dimension or aspect of a system, no more than one such aspectual evolution can be encoded and combined simultaneously. This is so because *Dynamic Logic Programming* is defined only for linear sequences of states. *Multi-dimensional Dynamic Logic Programming* ($\mathcal{MDLP}$) [10] was introduced to generalize *DLP* to allow for collections of states represented by arbitrary acyclic digraphs (*DAG*), not just sequences of states. $\mathcal{MDLP}$ assigns semantics to sets and subsets of logic programs, depending on how they stand in relation to one another, as defined by the *DAG* that represents the states and their configuration. By dint of such natural generalization, $\mathcal{MDLP}$ affords extra expressiveness, thereby enlarging the latitude of logic programming applications unifiable under a single framework. The flexibility provided by a *DAG* ensures a wide scope and variety of new possibilities. By virtue of the newly added characteristics of multiplicity and composition, $\mathcal{MDLP}$ provides a "societal" viewpoint in Logic Programming, important in these web and agent days, for combining knowledge in general.

In this paper we explore the application of the new paradigm of *Multi-dimensional Dynamic Logic Programming* to represent an agent's view of the combination of societal knowledge dynamics, i.e. the agent's view of the evolution of its knowledge as a result of knowledge evolution in the community of agents.

We begin with a brief overview of DLP in Section 2. In Section 3, we present $\mathcal{MDLP}$. In Section 4 we explore the application of $\mathcal{MDLP}$ to represent inter and intra-agent relationships and their views of a multi-agent system. We then conclude in Section 6.

2 Background

We start with an overview of the syntax and semantics of generalized logic programs, followed by a short recap of the paradigm of *Dynamic Logic Programming.*

2.1 Generalized Logic Programs and Their Stable Models

To represent *negative* information in logic programs and in their updates, since we need to allow default negation $not\, A$ not only in premises of their clauses but also in their heads, we use *generalized logic programs* as defined in [1][1]. By a *generalized logic program* P in a language $\mathcal{L}$ we mean a finite or infinite set of propositional clauses of the form $L_0 \leftarrow L_1, \ldots, L_n$ where each L_i is a literal (i.e. an atom A or the default negation of an atom $not\, A$). If r is a clause (or rule), by $H(r)$ we mean L, and by $B(r)$ we mean $L_1, \ldots, L_n$. If $H(r) = A$ (resp. $H(r) = not\, A$) then $not\, H(r) = not\, A$ (resp. $not\, H(r) = A$). By a (2-valued) *interpretation* M of $\mathcal{L}$ we mean any set of literals from $\mathcal{L}$ that satisfies the condition that for any A, *precisely one* of the literals A or $not\, A$ belongs to M. Given an interpretation M we define $M^+ = \{A : A \text{ is an atom}, A \in M\}$ and $M^- = \{not\, A : A \text{ is an atom}, \ not\, A \in M\}$. Following established tradition, wherever convenient we omit the default (negative) atoms when describing interpretations and models. We say that a (2-valued) interpretation M of $\mathcal{L}$ is a stable model of a generalized logic program P if $r(M) = least\,(r(P) \cup r(M^-))$, where $r(.)$ univocally renames every default literal $not\, A$ in a program or model into new atoms, say not_A. The class of generalized logic programs can be viewed as a special case of yet broader classes of programs, introduced earlier in [13], and, for the special case of normal programs, their semantics coincides with the stable models semantics [7].

[1] In [2] the reader can find the motivation for the usage of generalized logic programs, instead of using simple denials as a result of freely moving the head *not*s into the body.

2.2 Dynamic Logic Programming

Next we recall the semantics of *dynamic logic programming* [1]. A dynamic logic program is a sequence $P_0 \oplus ... \oplus P_n \oplus ...$ (also denoted by $\bigoplus\mathcal{P}$, where $\mathcal{P} = \{P_s : s \in S\}$ is a finite or infinite sequence of LPs, indexed by the finite or infinite set $S = \{1, 2, \ldots, n, \ldots\}$. Such sequence may be viewed as the outcome of updating P_0 with P_1, ..., updating it with P_n,... As we will see in the following sections, each P_i is determined by the i^{th} state transition. The role of dynamic logic programming is to ensure that these newly added rules are in force, and that previous rules are still valid (by inertia) for as long as they do not conflict with more recent ones, whenever the latter remain in force themselves. The notion of dynamic logic program at state s, denoted by $\bigoplus_s \mathcal{P}$, characterizes the meaning of the dynamic logic program when queried at state s, by means of its stable models, defined as follows:

Definition 1 (Stable Models of DLP). *Let* $\bigoplus\mathcal{P} = \bigoplus\{\ P_s : s \in S\}$ *be a* dynamic logic program, *let* $s \in S$. *An interpretation* M_s *is a* stable model of $\bigoplus\mathcal{P}$ at state s *iff* $M_s = least(\mathcal{P}_s - Reject(s, M_s) \cup Default(M_s))$ *where:*

$$\begin{array}{l}\mathcal{P}_s = \bigcup_{i \leq s} P_i \\ Reject(s, M_s) = \{r \in P_i : \exists r' \in P_j, i < j \leq s, H(r) = not\, H(r') \wedge M_s \vDash B(r')\} \\ Default(\mathcal{P}_s, M_s) = \{not\, A : \nexists r \in \mathcal{P}_s, H(r) = A \wedge M_s \vDash B(r)\}\end{array}$$

and where A is an atom.

3 Multi-dimensional Dynamic Logic Programming

Even though the main motivation behind the introduction of *DLP* was to represent the evolution of knowledge in time, the relationship between the different states can encode other aspects of a system, as pointed out in [1]. In fact, since its introduction, *DLP* (and *LUPS*) has been employed to represent a stock of features of a system, namely as a means to: represent and reason about the evolution of knowledge in time [1]; combine rules learnt by a diversity of agents [9]; reason about updates of agents' beliefs [5]; model agent interaction [15]; model and reason about actions [1]; resolve inconsistencies in metaphorical reasoning [12].

The common feature among these applications of *DLP* is that the states associated with the given set of theories encode only one of several possible representational dimensions (e.g. time, hierarchies, domains,...), inasmuch *DLP* is defined for linear sequences of states alone. For example, *DLP* can be used to model the relationship of a strict hierarchy group of agents, and *DLP* can be used to model the evolution of a single agent over time. But *DLP*, as it stands, cannot deal with both settings at once, and model the evolution of one such group of agents over time.

In effect, knowledge updating is not to be simply envisaged as taking place in the time dimension alone. Several updating dimensions may combine simultaneously, with or without the temporal one, such as specificity (as in taxonomies), strength of the updating instance (as in the legislative domain), hierarchical position of knowledge source (as in organizations), credibility of the source (as in uncertain, mined, or learnt knowledge), or opinion precedence (as in a society of agents). For this to be possible, *DLP* needs to be extended to allow for a more general structure of states.

In this section we present the notion of *Multi-dimensional Dynamic Logic Programming ($\mathcal{MDLP}$)* (introduced in [10]) which generalizes *DLP* to allow for collections of states represented by arbitrary acyclic digraphs. In this setting, $\mathcal{MDLP}$ assigns semantics to sets and subsets of logic programs, depending on how they relate to one another, these relations being defined by the acyclic digraph representing the states.

3.1 Graphs

A *directed graph*, or *digraph*, $D = (V, E)$ is a pair of two finite or infinite sets $V = V_D$ of *vertices* and $E = E_D$ of pairs of vertices or (*directed*) *edges*. A *directed edge sequence from* v_0 *to* v_n in a digraph is a sequence of edges $e_1, e_2, ..., e_n \in E_D$ such that $e_i = (v_{i-1}, v_i)$ for $i = 1, ..., n$. A *directed path* is a directed edge sequence in which all the edges are distinct. A *directed acyclic graph*, or *acyclic digraph (DAG)*, is a digraph D such that there are no directed edge sequences from v to v, for all vertices v of D. A *source* is a vertex with in-valency 0 (number of edges for which it is a final vertex) and a *sink* is a vertex with out-valency 0 (number of edges for which it is an initial vertex). We say that $v < w$ if there is a directed path from v to w and that $v \leq w$ if $v < w$ or $v = w$. The relevancy DAG of a DAG D wrt a vertex v of D is $D_v = (V_v, E_v)$ where $V_v = \{v_i : v_i \in V \text{ and } v_i \leq v\}$ and $E_v = \{(v_i, v_j) : (v_i, v_j) \in E \text{ and } v_i, v_j \in V_v\}$. The relevancy DAG of a DAG D wrt a set of vertices S of D is $D_S = (V_S, E_S)$ where $V_S = \bigcup_{v \in S} V_v$ and $E_S = \bigcup_{v \in S} E_v$, where $D_v = (V_v, E_v)$ is the relevancy DAG of D wrt v.

3.2 Declarative Semantics

We start by defining the framework consisting of the generalized logic programs indexed by a *DAG*. Throughout this paper, we will restrict ourselves to *DAG*'s such that for every vertex v of the *DAG*, any path ending in v is finite.

Definition 2 (Multi-dimensional Dynamic Logic Program). *Let $\mathcal{L}$ be a propositional language. A* Multi-dimensional Dynamic Logic Program (MDLP), *$\mathcal{P}$, is a pair $(\mathcal{P}_D, D)$ where $D = (V, E)$ is a DAG and $\mathcal{P}_D = \{P_v : v \in V\}$ is a set of generalized logic programs in the language $\mathcal{L}$, indexed by the vertices $v \in V$ of D. We call* states *such vertices of D. For simplicity, we often leave the language $\mathcal{L}$ implicit.*

To characterize the models of $\mathcal{P}$ at any given state we will keep to the basic intuition of logic program updates, whereby an interpretation is a stable model of the update of a program P by a program U iff it is a stable model of a program consisting of the rules of U together with a subset of the rules of P comprised by those that are not rejected (do not carry over by inertia) due to their being overridden by program U. With the introduction of a *DAG* to index programs, a program may have more than a single ancestor. This has to be dealt with, the desired intuition being that a program $P_v \in \mathcal{P}_D$ can be used to reject rules of any program $P_u \in \mathcal{P}_D$ if there is a directed path from u to v. Moreover, if some atom is not defined in the update nor in any of its ancestor, its negation is assumed by default. Formally, the stable models of the *MDLP* are:

Definition 3 (Stable Models at state s). *Let $\mathcal{P} = (\mathcal{P}_D, D)$ be a* MDLP*, where $\mathcal{P}_D = \{P_v : v \in V\}$, $D = (V, E)$ and $s \in V$. An interpretation M_s is a* stable model of $\mathcal{P}$ at state s *iff $M_s = least\left([\mathcal{P}_s - Reject(s, M_s)] \cup Default\left(\mathcal{P}_s, M_s\right)\right)$ where:*

$$\begin{array}{l}
\mathcal{P}_s = \bigcup_{i \le s} P_i \\
Reject(s, M_s) = \{r \in P_i \mid \exists r' \in P_j, i < j \le s, H(r) = not\, H(r') \wedge M_s \vDash B(r')\} \\
Default\left(\mathcal{P}_s, M_s\right) = \{not\, A \mid \nexists r \in \mathcal{P}_s : (H(r) = A) \wedge M_s \vDash B(r)\}
\end{array}$$

Intuitively, the set $Reject(s, M_s)$ contains those rules belonging to a program indexed by a state i that are overridden by the head of another rule with true body in state j along a path to state s. $\mathcal{P}_s$ contains all rules of all programs that are indexed by a state along all paths to state s, i.e. all rules that are potentially relevant to determine the semantics at state s. The set $Default\left(\mathcal{P}_s, M_s\right)$ contains default negations *not* A of all unsupported atoms A, i.e., those atoms A for which there is no rule in $\mathcal{P}_s$ whose body is true in M_s.

According to [10], to determine the models of a *MDLP* at state s, we need only consider the part of the *MDLP* corresponding to the relevancy graph wrt state s.

We might have a situation where we desire to determine the semantics jointly at more than one state. If all these states belong to the relevancy graph of one of them, we simply determine the models at that state. But this might not be the case. Formally, the semantics of a *MDLP* at an arbitrary set of its states is determined by the definition:

Definition 4 (Stable Models at a set of states S). *Let $\mathcal{P} = (\mathcal{P}_D, D)$ be a* MDLP*, where $\mathcal{P}_D = \{P_v : v \in V\}$ and $D = (V, E)$. Let S be a set of states such that $S \subseteq V$. An interpretation M_S is a* stable model of $\mathcal{P}$ at the set of states S *iff $M_S = least\left([\mathcal{P}_S - Reject(S, M_S)] \cup Default\left(\mathcal{P}_S, M_S\right)\right)$ where:*

$$\begin{array}{l}
\mathcal{P}_S = \bigcup_{s \in S} \left(\bigcup_{i \le s} P_i\right) \\
Reject(S, M_S) = \left\{ \begin{array}{c} r \in P_i \mid \exists s \in S, \exists r' \in P_j, i < j \le s, \\ H(r) = not\, H(r') \wedge M_S \vDash B(r') \end{array} \right\} \\
Default\left(\mathcal{P}_S, M_S\right) = \{not\, A \mid \nexists r \in \mathcal{P}_S : (H(r) = A) \wedge M_S \vDash B(r)\}
\end{array}$$

This is equivalent to the addition of a new vertex α to the DAG, and connecting to α, by addition of edges, all states we wish to consider. Furthermore, the program indexed by α is empty. We then determine the stable models of this new *MDLP* at state α. Note the addition of state α does not affect the stable models at other states. Indeed, α and the newly introduced edges do not belong to the relevancy DAG wrt any other state. A particular case of the above definition is when $S = V$, corresponding to the semantics of the whole *MDLP*. In [10], we have presented an alternative definition, based on a purely syntactical transformation that, given a *MDLP*, produces a generalized logic program whose stable models are in a one-to-one equivalence relation with the stable models of the *MDLP* previously characterized. The computation of the stable models at some state s reduces to the computation of the transformation followed by the computation of the stable models of the transformed program. This directly provides for an implementation of $\mathcal{MDLP}$, publicly available at `centria.di.fct.unl.pt/~jja/updates`.

4 Inter- and Intra-Agent Social Viewpoints

The previous section contains the definition of the notion of *Multi-dimensional Dynamic Logic Programming*, $\mathcal{MDLP}$, as an extension of *DLP* to allow for states to be related by an arbitrary DAG. The stable models of $\mathcal{MDLP}$ have been characterized but nothing has been yet explained as how to use such DAGs to represent real problems. In particular, we have not shown how DAGs allow for the combination of more than one representational dimension, the very motivation to introduce $\mathcal{MDLP}$. Here, we explore some particular classes of DAGs suitable in the context of multi-agent systems.

Agents are situated and therefore need to represent and reason about information they obtain directly by sensing the environment or communicated by other agents. These agents, as well as the environment, evolve in time, i.e. the incoming information is to be used as an update over existing knowledge. Moreover these agents do not have the same credibility, this being represented via a hierarchy of predominance. In this section we explore DAGs that provide a way to represent the evolution in time of knowledge with provenance in a community of hierarchically related agents.

We start with an agent α, situated in a community of agents represented by the greek letters β, γ, μ, ν. The multi-agent system is $\mathcal{A} = \{\alpha, \beta, \gamma, \mu, \nu\}$. According to agent α's hierarchical view of the world, and its position within the community, all agents are related according to the DAG $D_h = (\mathcal{A}, E_h)$ where $E_h = \{(\nu, \mu), (\beta, \mu), (\mu, \gamma), (\mu, \alpha), (\gamma, \alpha)\}$, depicted in Fig. 1 a).

According to this DAG, agent α's opinions prevail over those of every other agent. However this need not be so. If, for example, one of these agent's role was to coordinate the community, it would be natural to exist an edge connecting α to this agent.

In a static environment, this representation would be sufficient to determine the semantics of α's view of the community. In such a situation, the rules asserted by each agent would constitute programs indexed by the DAG of Fig. 1 a), i.e. $P_\beta, P_\gamma, P_\mu, ...$

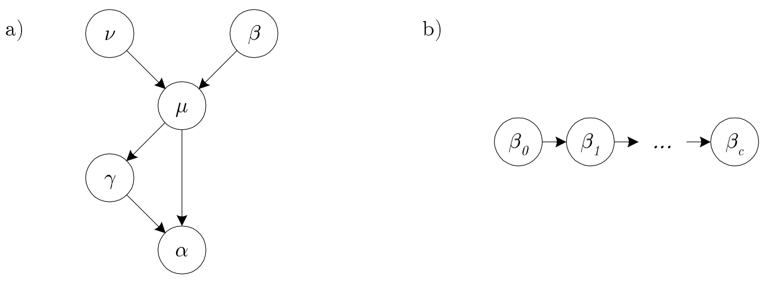


Fig. 1. a) Hierarchical Dimension b) Temporal Dimension

In a realistic scenario, where the dynamics of the system cannot be ignored, there is no single program representing each agent. Rather, there is a sequence of programs representing the knowledge of each agent at each time point. Suppose these time points were represented by the set $T = \{0, 1, ..., c\}$ (where by c we mean the current time state), then, for example, the knowledge of agent β would be represented by the set of programs $\{P_{\beta_0}, P_{\beta_1}, ..., P_{\beta_c}\}$, indexed according to the DAG $D_{\beta_t} = (B_t, E_t)$ where $B_t = \{\beta_t : t \in T\}$ and $E_t = \{(0,1), ..., (c-1, c)\}$ as depicted in Fig. 1 b).

The full dynamic hierarchical scenario, comprising all agents, is then represented by the set of programs $\mathcal{P}_D = \{P_{a_t} : a \in \mathcal{A}, t \in T\}$ indexed by the DAG $D = (\mathcal{A}_T, E)$ where $\mathcal{A}_T = \{a_t : a \in \mathcal{A}, t \in T\}$.

There still remains to be defined the relationships between all these programs, i.e. the edges belonging to E. To this purpose, we will propose three basic ways to systematically relate these programs.

4.1 Equal Role Representation

The first approach to combining the hierarchical and temporal dimensions is accomplished by assigning equal roles to both precedence relations. In this scenario, we maintain the temporal precedence relation within each agent, and the hierarchical one within each time state, and we do not relate any two programs that fall outside this scope, i.e. there is no precedence between a higher ranked older program and a lower ranked newer one. Accordingly, the set of edges E, of the DAG D contains the union of the following two sets of edges:

Time Dependence Edges (TDE) : $\{(a_{t_1}, a_{t_2}) : a \in \mathcal{A}, t_1, t_2 \in T, t_1 < t_2\}$.
Hierarchy Dependence Edges (HDE) : $\{(a_t, b_t) : a, b \in \mathcal{A}, t \in T, a < b\}$.

Intuitively, each rule can be used to reject any rule of a lower ranked agent indexed by a time state equal or lower than its own. This situation is depicted in Fig. 2.

Remark 1. Throughout this section, we have chosen a simplified representation of the DAGs to make their interpretation easier. For this purpose, we introduce

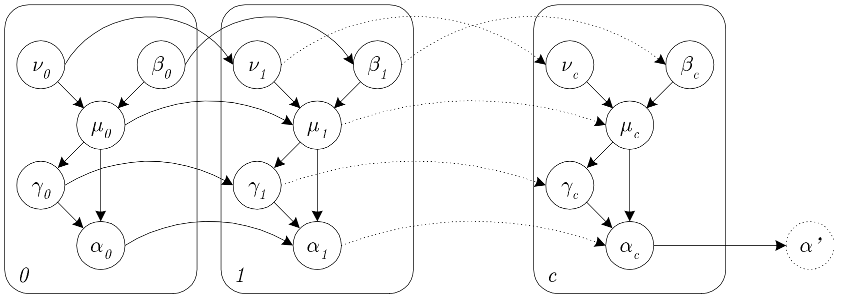


Fig. 2. Equal Role Representation

new nodes (meta-nodes) encapsulating part of the DAG (detail). To obtain the complete DAG from this simplification one needs to replace the meta-node with the detail while replacing the edges entering the meta-node with a set of edges entering each source node of the detail. Similarly, one needs to replace each node departing from the meta-node with a set of edges departing from each sink of the detail. In every DAG, we have added a new node labelled α', which becomes its single sink, and an empty program associated with it, indicating where the semantics corresponding to agent α's view of the overall system at time state c can be determined. Also, since the semantics of MDLP is invariant wrt the transitive closure of the DAG, we will often be omitting some edges that do not affect such transitive closure.

Such a scenario can be found in legal reasoning, where the legislative agency is divided conforming to a hierarchy of power, governed by the principle *Lex Superior (Lex Superior Derogat Legi Inferiori)* according to which the rule issued by a higher hierarchical authority overrides the one issued by a lower one, and the evolution of law in time is governed by the principle *Lex Posterior (Lex Posterior Derogat Legi Priori)* according to which the rule enacted at a later point in time overrides the earlier one. *Lex Superior* is encoded by the Hierarchy Dependence Edges and *Lex Posterior* is encoded the Time Dependence Edges.

Allowing rejection governed by time and hierarchy alone, potentiates contradiction inasmuch as there are many pairs of programs not related according to this graph. If the purpose of our agency system were to perform some sort of paraconsistent reasoning, such as in an agent based negotiation system trying to reach an agreement, this would be the ideal scenario: contradiction would generate messages to the responsible agents to possibly review their positions. But often this is not the case and we may want to reduce the amount of contradiction, namely by establishing a skewed relation between the temporal and hierarchical dimensions. Two approaches will be explored in the following subsections.

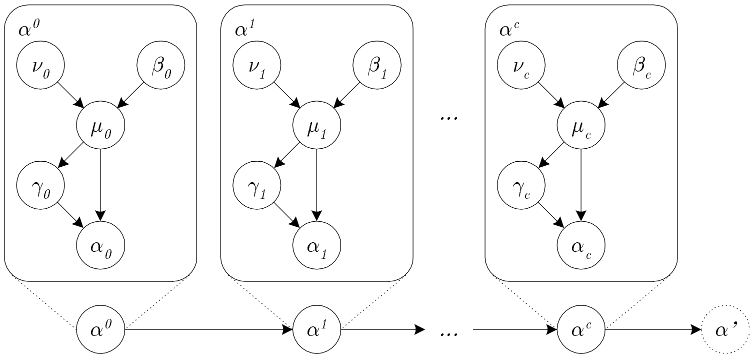


Fig. 3. Time Prevailing Representation

4.2 Time Prevailing Representation

According to this representation, the DAG D contains, besides the Time and Hierarchy Dependence Edges, the following edges:

Time Prevailing Edges (TPE) $: \{(a_{t_1}, b_{t_2}) : a, b \in \mathcal{A}, t_1, t_2 \in T, t_1 < t_2\}$.

The intuitive reading is that any rule indexed by a more recent time state overrides any older rule, independently of which agents these rules belong to. This situation is depicted in Fig. 3.

This representation is particularly useful in very dynamic situations where competence is distributed, i.e. when knowledge changes rapidly and different agents will typically provide rules about different literals. This is so mainly because any newer rule always overrides any older one. It means that if a situation is completely defined by the rules issued by the community at a given time state, one can simply ignore older rules.

The main drawback of this representation relates to the trustfulness of agents in the community. It requires all agents to be fully trusted because, in allowing all new rules to override all old ones, irrespective of their hierarchical position, any untrustworthy lower ranked agent can override any higher ranked agent just by issuing a rule at a later time state. This leads us to the next, alternative, representation.

4.3 Hierarchy Prevailing Representation

According to this representation, the DAG D contains, besides the Time and Hierarchy Dependence Edges, the following edges:

Hierarchy Prevailing Edges (HPE) $: \{(a_{t_1}, b_{t_2}) : a, b \in \mathcal{A}, t_1, t_2 \in T, a < b\}$.

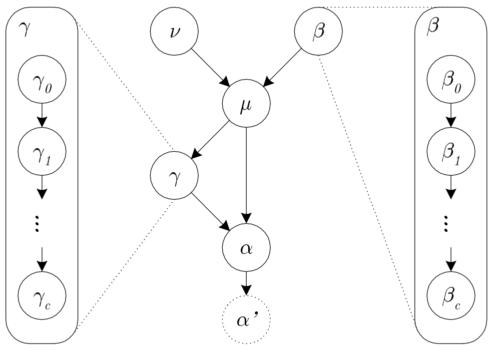


Fig. 4. Hierarchy Prevailing Representation

The intuitive reading is that any rule indexed by a higher ranked agent overrides any lower ranked agent's rule, independently of the time state it is indexed by. This situation is depicted in Fig. 4.

This situation is useful, in contrast with the previous one, when some of the agents are untrustworthy because a lower ranked agent rule, to be used, may not be contradicted by any (even if older) higher ranked agent rule. The main drawback is that one has to consider the entire history of all higher ranked agents in order to accept/reject a rule provided by a lower ranked agent. However, a number of techniques to reduce the size of a dynamic logic program are being developed, useful for simplifying the time sequence of programs of each individual agent. These are outside the scope of this paper.

Again in the context of Legal Reasoning, this scenario corresponds to the one used in many Legislatures, where collisions between rules are governed by the principle *Lex Superior Priori Derogat Legi Inferiori Posterior*, i.e. the rule issued by a higher hierarchical authority at an earlier point overrides the one issued by a lower hierarchical authority at a later point.

4.4 Representing Inter- and Intra-Agent Relationships

The representations set forth in the previous sub-sections refer to a community of agents. Nevertheless, they can be used at different levels of abstraction to represent macro and micro aspects of a multi-agent system, in a unified manner. Let us suppose that agent α is composed of several sub-agents concurrently performing dedicated tasks while reading and writing onto a common knowledge structure. According to this view, agent α can now be seen as a community of sub-agents $\mathcal{A}_\alpha = \{\alpha_a, \alpha_b, \alpha_d, \alpha_e\}$, related, for example, according to the DAG $D_\alpha = (\mathcal{A}_\alpha, E_\alpha)$ where $E_\alpha = \{(\alpha_a, \alpha_b), (\alpha_a, \alpha_e), (\alpha_b, \alpha_d), (\alpha_e, \alpha_d)\}$ as in Fig. 5. The overall dynamic system, comprising all agents and sub-agents, is now represented by the set of programs $\mathcal{P}_D = \{P_{a_t} : a_t \in \mathcal{A}_T\}$ indexed by the DAG $D = (\mathcal{A}_T, E)$ where $\mathcal{A}_T = \{a_t : a \in \mathcal{A} \setminus \{\alpha\}, t \in T\} \cup \{a_t : a \in \mathcal{A}_\alpha, t \in T\}$.

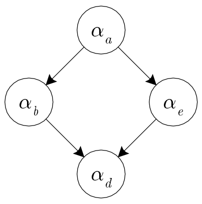


Fig. 5. Sub-agent Hierarchy

As for the relations between the programs, we propose a combination of the time and hierarchy prevailing representations to relate the sub-agents and agents respectively. As mentioned before, the time prevailing representation is the most efficient but requires all agents to be trusted. One would expect an agent to trust its component sub-agents. As for the representation of other agents, we will opt for the hierarchy prevailing relation. Formally, the set of edges in the DAG contains:

Time Prevailing Edges (TPE) : $\{(a_{t_1}, b_{t_2}) : a, b \in \mathcal{A}_\alpha, t_1, t_2 \in T, t_1 < t_2\}$, to model the relationships between the sub-agents of α.

Hierarchy Prevailing Edges (HPE) : $\{(a_{t_1}, b_{t_2}) : a, b \in \mathcal{A}, t_1, t_2 \in T, a < b\}$, to model the relationships between the agents of the system. Note that each edge entering (resp. departing from) α_t should be interpreted as a set of edges entering (resp. departing from) each of $\{\alpha_{a_t}, \alpha_{b_t}, \alpha_{d_t}, \alpha_{e_t}\}$.

This situation is depicted in Fig. 6. Note however that this is just one proposal of the many possible existing combinations to represent such relations.

5 Conclusions

In this paper we have explored Multi-dimensional Dynamic Logic Programming as a means to combine knowledge provenient from different agents, into a single knowledge base point of view, with a precise declarative semantics. Depending on the situation and the relations amongst the agents, we have envisaged several classes of acyclic digraphs suitable for its encoding.

Based on the strengths of $\mathcal{MDLP}$ as a framework capable of simultaneously represent several aspects of a system in a dynamic fashion, and of *LUPS* [2] as a powerful language to specify the evolution of such representations by means of transitions, we have launched into the design of an agent architecture, $\mathcal{MINERVA}$ [11]. It aims at providing, on a sound theoretical basis, a common agent framework based on the strengths of Logic Programming, to allow the combination of several non-monotonic knowledge representation and reasoning mechanisms developed in recent years.

The use of Logic Programming for the overall endeavour is justified on the ground of it providing a rigorous single encompassing theoretical basis for the

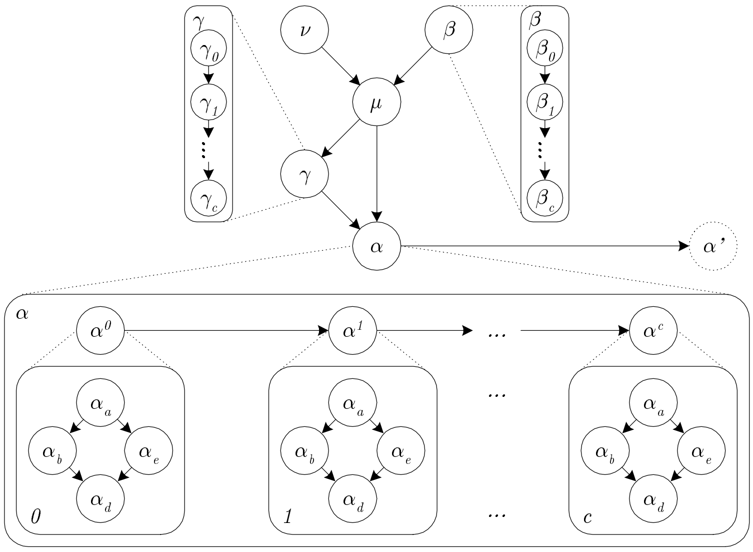


Fig. 6. Inter- and Intra-Agent Relationship Representation

aforesaid topics, as well as an implementation vehicle for parallel and distributed processing. Additionally, logic programming provides a formal high level flexible instrument for the rigorous specification and experimentation with computational designs, making it extremely useful for prototyping, even when other, possibly lower level, target implementation languages are envisaged.

Rational agents, in our opinion, will require an admixture of any number of those reasoning mechanisms mentioned in the introduction, to carry out their tasks. To this end, a $\mathcal{MINERVA}$ agent is based on a modular design where a common knowledge base is concurrently manipulated by specialized sub-agents. The common knowledge base contains all knowledge shared by more than one sub-agent. It is conceptually divided in the following components: *Capabilities*, *Intentions*, *Goals*, *Plans*, *Reactions*, *Object Knowledge Base* and *Internal Behaviour Rules.* Although conceptually divided in such components, all these modules share a common representation mechanism based on $\mathcal{MDLP}$ and *LUPS*, the former to represent knowledge at each state and *LUPS* to represent the state transitions, i.e. the common part of the agent's behaviour. Every agent is composed of specialized functionality related subagents, that execute various specialized tasks. Examples of such subagents are those implementing the reactive, planning, scheduling, belief revision, goal management, learning, dialogue management, information gathering, preference evaluation, strategy, and diagnosis functionalities. These sub-agents contain a *LUPS* program encoding their behaviour, and interfacing with the *Common Knowledge Base.* Whilst some of those sub-agent's functionalities are fully specifiable in *LUPS*, others require private specialized procedures where *LUPS* serves as an interface language.

References

1. J. J. Alferes, J. A. Leite, L. M. Pereira, H. Przymusinska, and T. Przymusinski. Dynamic updates of non-monotonic knowledge bases. *Journal of Logic Programming*, 45(1-3):43–70, 2000. Short version titled *Dynamic Logic Programming* appeared in Procs. of KR-98.
2. J. J. Alferes, L. M. Pereira, H. Przymusinska, and T. Przymusinski. LUPS : A language for updating logic programs. *Artificial Intelligence*, 2001. To appear. Short version appeared in Procs of LPNMR-99, LNAI-1730.
3. M. Bozzano, G. Delzanno, M. Martelli, V. Mascardi, and F. Zini. Logic programming and multi-agent system: A synergic combination for applications and semantics. In *The Logic Programming Paradigm - A 25-Year Perspective*. Springer, 1999.
4. F. Buccafurri, W. Faber, and N. Leone. Disjunctive logic programs with inheritance. In *Procs. of ICLP-99*. MIT Press, 1999.
5. P. Dell'Acqua and L. M. Pereira. Updating agents. In *Procs of MAS-99, ICLP-99 Ws.*, 1999.
6. T. Eiter, M. Fink, G. Sabbatini, and H. Tompits. Considerations on updates of logic programs. In *Procs. of JELIA-00*, LNAI-1919. Springer, 2000.
7. M. Gelfond and V. Lifschitz. The stable semantics for logic programs. In *Procs. of ICLP-88*. MIT Press, 1988.
8. N. R. Jennings, K. Sycara, and M. Wooldridge. A roadmap of agent research and development. *Journal of Autonomous Agents and Multi-Agent Systems*, 1(1):7–38, 1998.
9. E. Lamma, F. Riguzzi, and L. M. Pereira. Strategies in combined learning via logic programs. *Machine Learning*, 38(1/2):63–87, 2000.
10. J. A. Leite, J. J. Alferes, and L. M. Pereira. Multi-dimensional dynamic logic programming. In *Procs. of CLIMA'00*, pages 17–26, 2000.
11. J. A. Leite, J. J. Alferes, and L. M. Pereira. $\mathcal{MINERVA}$ - a dynamic logic programming agent architecture. In *Procs. of ATAL'01*, 2001.
12. J. A. Leite, F. C. Pereira, A. Cardoso, and L. M. Pereira. Metaphorical mapping consistency via dynamic logic programming. In *Procs. of AISB'00*. AISB, 2000.
13. V. Lifschitz and T. Woo. Answer sets in general non-monotonic reasoning (preliminary report). In *Procs. of KR-92*. Morgan-Kaufmann, 1992.
14. I. Niemelä and P. Simons. Smodels: An implementation of the stable model and well-founded semantics for normal LP. In *Procs. of LPNMR'97*, volume 1265 of *LNAI*. Springer, 1997.
15. P. Quaresma and I. P. Rodrigues. A collaborative legal information retrieval system using dynamic logic programming. In *Procs. of ICAIL-99*. ACM Press, 1999.
16. F. Sadri and F. Toni. Computational logic and multiagent systems: A roadmap, 1999. Available from `http://www.compulog.org`.
17. C. Sakama and K. Inoue. Updating extended logic programs through abduction. In *Procs. of LPNMR-99*. Springer, 1999.
18. XSB-Prolog. The XSB logic programming system, version 2.0, 1999. Available at `http://www.cs.sunysb.edu/ sbprolog`.
19. Y. Zhang and N. Foo. Updating logic programs. In *Procs. of ECAI'98*. Morgan Kaufmann, 1998.

A Procedural Semantics for Multi-adjoint Logic Programming

Jesús Medina[1], Manuel Ojeda-Aciego[1], and Peter Vojtáš[2]

[1] Dept. Matemática Aplicada. Universidad de Málaga.[⋆⋆⋆]
{jmedina,aciego}@ctima.uma.es
[2] Dept. Mathematical Informatics. P.J. Šafárik University.[†]
vojtas@kosice.upjs.sk

Abstract. Multi-adjoint logic program generalise monotonic logic programs introduced in [1] in that simultaneous use of several implications in the rules and rather general connectives in the bodies are allowed. In this work, a procedural semantics is given for the paradigm of multi-adjoint logic programming and completeness theorems are proved.

1 Introduction

Multi-adjoint logic programming was introduced in [5] as a refinement of both initial work in [7] and residuated logic programming [1]. It allows for very general connectives in the body of the rules, and sufficient conditions for the continuity of its semantics are known. Such an approach is interesting for applications: in [6] a system is presented where connectives are learnt from different users' examples and, thus, one can imagine a scenario in which knowledge is described by a many-valued logic program where connectives have many-valued truth functions and @ is an aggregation operator (arithmetic mean, weighted sum, ...) where different implications could be needed for different purposes, and different aggregators are defined for different users, depending on their preferences, e.g. the rule below:

$$\begin{aligned}&\texttt{hotel\_reservation(Business\_Location, Time, Hotel)} \longleftarrow_i\\&\qquad @(\texttt{near\_to}(\texttt{Business\_Location},\texttt{Hotel}),\\&\qquad\quad \texttt{cost\_reasonable}(\texttt{Hotel},\texttt{Time}),\\&\qquad\quad \texttt{building\_is\_fine}(\texttt{Hotel})).\ \texttt{with truth value}\ 0.8\end{aligned}$$

The framework of multi-adjoint logic programming was introduced in [5] as a generalisation of the monotonic and residuated logic programs given in [1]. The special features of multi-adjoint logic programs are: (1) a number of different implications are allowed in the bodies of the rules, (2) sufficient conditions for continuity of its semantics are known, and (3) the requirements on the lattice of truth-values are weaker that those for the residuated approach.

⋆⋆⋆ Partially supported by Spanish DGI project BFM2000-1054-C02-02 and Junta de Andalucía project TIC-115.
† Partially supported by Slovak project VEGA 1/7557/20

The purpose of this work is to provide a procedural semantics to the paradigm of multi-adjoint logic programming. This work is an extension of [3], with a different treatment of reductants and an improved version of the completeness results, based on the so-called supremum property.

The central topics of this paper are mainly at the theoretical level, however the obtained results can be applied in a number of contexts:

1. The integration of information retrieval and database systems requires methods for dealing with uncertainty; there are already a number of many-valued approaches to the general theory of databases, but none of them contains a formal mathematical proof of the relation between the relational algebra and its denotational counterpart; a by-product of the obtained results is the possibility of defining a fuzzy relational algebra and a fuzzy Datalog, the completeness result then shows that the expressive power of fuzzy Datalog is the same that the computational power of the fuzzy relational algebra;
2. One of the problems of fuzzy knowledge bases is handling a great amount of items with very small confidence value. The approach introduced in this paper enables us to propose a sound and complete threshold computation model oriented to the best correct answers up to a prescribed tolerance level.
3. The multi-adjoint framework can also be applied to abduction problems. In [4] the possibility of obtaining the cheapest possible explanation to an abduction problem wrt a cost function by means of a logic programming computation followed by a linear programming optimization has been shown.

2 Preliminary Definitions

To make this paper as self-contained as possible, the necessary definitions about multi-adjoint structured are included in this section. For motivating comments, the interested reader is referred to [5].

Definition 1. *Let $\langle L, \preceq \rangle$ be a complete lattice. A* multi-adjoint lattice *$\mathcal{L}$ is a tuple $(L, \preceq, \leftarrow_1, \&_1, \dots, \leftarrow_n, \&_n)$ satisfying the following items:*

1. $\langle L, \preceq \rangle$ is bounded, i.e. it has bottom and top elements;
2. $\top \&_i \vartheta = \vartheta \&_i \top = \vartheta$ for all $\vartheta \in L$ for $i = 1, \dots, n$;
3. $(\leftarrow_i, \&_i)$ is an adjoint pair in $\langle L, \preceq \rangle$ for $i = 1, \dots, n$; i.e.
   (a) Operation $\&_1$ is increasing in both arguments,
   (b) Operation $\leftarrow_i$ is increasing in the first argument and decreasing in the second argument,
   (c) For any $x, y, z \in P$, we have that $x \preceq (y \leftarrow_i z)$ holds if and only if $(x \&_i z) \preceq y$ holds.

From the point of view of expressiveness, it is interesting to allow extra operators to be involved with the operators in the multi-adjoint lattice. The structure which captures this possibility is that of a *multi-adjoint Ω-algebra* which can be understood as an extension of a multi-adjoint lattice containing a number of extra operators given by a signature Ω.

We will be working with two Ω-algebras: the first one, $\mathfrak{F}$, to define the syntax of our programs, and the second one, $\mathfrak{L}$, to host the manipulation of the truth-values of the formulas in the programs. To avoid possible name-clashes, we will denote the interpretation of an operator symbol ω in Ω under $\mathfrak{L}$ as $\dot{\omega}$ (a dot on the operator), whereas ω itself will denote its interpretation under $\mathfrak{F}$.

Definition 2 (Multi-Adjoint Logic Programs). *A* multi-adjoint logic program on a multi-adjoint Ω-algebra $\mathfrak{F}$ with values in a multi-adjoint lattice $\mathfrak{L}$ *(in short* multi-adjoint program*) is a set $\mathbb{P}$ of rules of the form $\langle (A \leftarrow_i \mathcal{B}), \vartheta \rangle$.*

1. *The* rule $(A \leftarrow_i \mathcal{B})$ *is a formula of $\mathfrak{F}$;*
2. *The* confidence factor ϑ *is an element (a truth-value) of L;*
3. *The* head *of the rule A is a propositional symbol of Π.*
4. *The* body *formula $\mathcal{B}$ is a formula of $\mathfrak{F}$ built from propositional symbols $B_1, \dots, B_n$ $(n \geq 0)$ by the use of conjunctors $\&_1, \dots, \&_n$ and $\wedge_1, \dots, \wedge_k$, disjunctors $\vee_1, \dots, \vee_l$ and aggregators $@_1, \dots, @_m$.*
5. Facts *are rules with body $\top$.*
6. *A* query *(or* goal*) is a propositional symbol intended as a question $?A$ prompting the system.*

As usual, an *interpretation* is a mapping $I \colon \Pi \to L$. The set of all interpretations of the formulas defined by the Ω-algebra $\mathfrak{F}$ in the Ω-algebra $\mathfrak{L}$ is denoted $\mathcal{I}_{\mathfrak{L}}$. Note that each of these interpretations can be uniquely extended to the whole set of formulas, $\hat{I} \colon F_{\Omega} \to L$.

The ordering $\preceq$ of the truth-values L can be easily extended to $\mathcal{I}_{\mathfrak{L}}$, which also inherits the structure of complete lattice.

Definition 3.

1. *An interpretation $I \in \mathcal{I}_{\mathfrak{L}}$* satisfies $\langle A \leftarrow_i \mathcal{B}, \vartheta \rangle$ *if and only if $\vartheta \preceq \hat{I}\,(A \leftarrow_i \mathcal{B})$.*
2. *An interpretation $I \in \mathcal{I}_{\mathfrak{L}}$ is a* model *of a multi-adjoint logic program $\mathbb{P}$ iff all weighted rules in $\mathbb{P}$ are satisfied by I.*
3. *An element $\lambda \in L$ is a* correct answer *for a program $\mathbb{P}$ and a query $?A$ if for any interpretation $I \in \mathcal{I}_{\mathfrak{L}}$ which is a model of $\mathbb{P}$ we have $\lambda \preceq I(A)$.*

The immediate consequences operator, given by van Emden and Kowalski, can be easily generalised to the framework of multi-adjoint logic programs.

Definition 4. *Let $\mathbb{P}$ be a multi-adjoint program. The* immediate consequences operator $T^{\mathfrak{L}}_{\mathbb{P}} \colon \mathcal{I}_{\mathfrak{L}} \to \mathcal{I}_{\mathfrak{L}}$*, mapping interpretations to interpretations, is defined by*

$$T^{\mathfrak{L}}_{\mathbb{P}}(I)(A) = \sup \left\{ \vartheta \mathbin{\dot{\&}_i} \hat{I}(\mathcal{B}) \mid A \overset{\vartheta}{\leftarrow}_i \mathcal{B} \in \mathbb{P} \right\}$$

As usual, the semantics of a multi-adjoint logic program is characterised by the post-fixpoints of $T^{\mathfrak{L}}_{\mathbb{P}}$, see [5]; that is, an interpretation I of $\mathcal{I}_{\mathfrak{L}}$ is a model of a multi-adjoint logic program $\mathbb{P}$ iff $T^{\mathfrak{L}}_{\mathbb{P}}(I) \sqsubseteq I$. It is remarkable the fact that this result is still true even without any further assumptions on conjunctors (definitely they need not be commutative and associative).

Regarding continuity, the following theorem was proved in [5].

Theorem 1 **([5]).**

1. *If all the operators occurring in the bodies of the rules of a program $\mathbb{P}$ are continuous, and the adjoint conjunctions are continuous in their second argument, then $T^{\mathfrak{L}}_{\mathbb{P}}$ is continuous.*
2. *If the operator $T^{\mathfrak{L}}_{\mathbb{P}}$ is continuous for all program $\mathbb{P}$ on $\mathfrak{L}$, then any operator in the body of the rules is continuous.*

3 Procedural Semantics of Multi-adjoint Logic Programs

Once we have shown that the $T^{\mathfrak{L}}_{\mathbb{P}}$ operator can be continuous under very general hypotheses, then the least model can be reached in at most countably many iterations. Therefore, it is worth to define a procedural semantics which allow us to actually construct the answer to a query against a given program.

For the formal description of the computational model, we will consider an extended the language $\mathfrak{F}'$ defined on the same graded set, but whose carrier is the disjoint union $\Pi \uplus L$; this way we can work simultaneously with propositional symbols and with the truth-values they represent.

Definition 5. *Let $\mathbb{P}$ be a multi-adjoint program, and let $V \subset L$ be the set of truth values of the rules in $\mathbb{P}$. The* extended language $\mathfrak{F}'$ *is the corresponding Ω-algebra of formulas freely generated from the disjoint union of Π and V.*

We will refer to the formulas in the language $\mathfrak{F}'$ simply as *extended formulas.* An operator symbol ω interpreted under $\mathfrak{F}'$ will be denoted as $\bar{\omega}$.

Our computational model will take a query (atom), and will provide a lower bound of the value of A under any model of the program. Given a program $\mathbb{P}$, we define the following admissible rules for transforming any extended formula.

Definition 6. Admissible rules *are defined as follows:*

1. *Substitute an atom A in an extended formula by $(\vartheta \bar{\&}_i \mathcal{B})$ whenever there exists a rule $\langle A \leftarrow_i \mathcal{B}, \vartheta\rangle$ in $\mathbb{P}$.*
2. *Substitute an atom A in an extended formula by $\bot$.*
3. *Substitute an atom A in an extended formula by ϑ whenever there exists a fact $\langle A \leftarrow_i \top, \vartheta\rangle$ in $\mathbb{P}$.*

Note that if an extended formula turns out to have no propositional symbols, then it can be directly interpreted in the multi-adjoint Ω-algebra $\mathfrak{L}$. This justifies the following definition of *computed answer.*

Definition 7. *Let $\mathbb{P}$ be a multi-adjoint program, and let $?A$ be a goal. An element $\dot{@}[r_1, \dots, r_m]$, with $r_i \in L$, for all $i \in \{1, \dots, m\}$ is said to be a* computed answer *if there is a sequence $G_0, \dots, G_{n+1}$ such that*

1. *$G_0 = A$ and $G_{n+1} = \bar{@}[r_1, \dots, r_m]$ where $r_i \in L$ for all $i = 1, \dots n$.*
2. *Every G_i, for $i = 1, \dots, n$, is a formula in $\mathfrak{F}'$.*
3. *Every G_{i+1} is inferred from G_i by one of the admissible rules.*

Note that our procedural semantics, instead of being refutation-based (this is not possible, since negation is not allowed in our approach), is oriented to obtaining a bound of the optimal correct answer of the query.

3.1 Reductants

It might be the case that for some lattices it is not possible to get the correct answer, simply consider L to be the powerset of a two-element set $\{a, b\}$ ordered by inclusion, and the following example from Morishita, used in [2]:

Example 1. Given a multi-adjoint program $\mathbb{P}$ with rules $A \overset{a}{\leftarrow} B$ and $A \overset{b}{\leftarrow} B$ and fact $(B, \top)$. Assuming that the adjoint conjunction to $\leftarrow$ has the usual boundary conditions, then the correct answer to the query $?A$ is $\top$, since it has to be an upper bound of all the models of the program, therefore it has to be greater than both a and b. But the only computed answers are either a or b. $\square$

The idea to cope with this problem is the generalisation of the concept of reductant [2] to our framework. Namely, that whenever we have a finite number of rules $A \overset{\vartheta_i}{\leftarrow}_i @_i(D^i_1, \dots, D^i_{n_i})$ for $i = 1, \dots, k$, then there should exist another rule which allows us to get the correct value of A under the program. That can be rephrased as follows:

As any rule $A \overset{\vartheta_i}{\leftarrow}_i @_i(D_1, \dots, D_{n_i})$ contributes with the value $\vartheta_i \mathbin{\dot{\&}_i} b_i$ for the calculation of the lower bound for the truth-value of A, we would like to have the possibility of reaching the supremum of all the contributions, in the computational model, in a single step. This leads to the following definition.

Definition 8 (Reductant). *Let $\mathbb{P}$ be a multi-adjoint program; assume that all the rules in $\mathbb{P}$ with head A are $A \overset{\vartheta_i}{\leftarrow}_i \mathcal{B}_i$ for $i = 1, \dots, n$. A* reductant for A *is a rule $A \overset{\vartheta}{\leftarrow} @(\mathcal{B}_1, \dots, \mathcal{B}_n)$ such that for any $b_1, \dots, b_n$ we have*

$$\sup\{\vartheta_i \mathbin{\dot{\&}_i} b_i \mid i = 1, \dots, n\} = \vartheta \mathbin{\dot{\&}} \dot{@}(b_1, \dots, b_n)$$

where $\&$ is the adjoint conjunctor to the implication $\leftarrow$.

Remark 1. When all the elements in the multiset are the same (e.g. all rules in the program have the same implication), and the operator $\dot{\&}$ is continuous, then the following equality holds

$$\sup\{\vartheta_i \mathbin{\dot{\&}} b_i \mid i = 1, \dots, n\} = \sup \vartheta_i \mathbin{\dot{\&}} \sup b_i$$

which immediately implies that choosing $\vartheta = \sup \vartheta_i$ and $\dot{@}(b_1, \dots, b_n) = \sup b_i$ we have constructed a reductant.

The example below shows a program with reductants, whose truth-values range over the lattice of closed intervals in $[0, 1]$.

Example 2. Consider the lattice of all the closed intervals in $[0, 1]$, denoted $\mathcal{C}[0, 1]$ under the ordering $[a, b] \preceq [c, d]$ iff $a \leq c$ and $d \leq b$, and consider the componentwise extended definition of the Łukasiewicz, product and Gödel implications to intervals. Let $\mathbb{P}$ be the program with rules

$$\langle A \longleftarrow_P B, [\vartheta_1, \vartheta_2] \rangle \qquad \langle A \longleftarrow_G C, [\tau_1, \tau_2] \rangle$$

where $\vartheta_1 \leq \tau_1$, and facts

$$\langle B \longleftarrow_L, [\vartheta_3, \vartheta_4] \rangle \qquad\qquad \langle C \longleftarrow_P, [\vartheta_5, \vartheta_6] \rangle$$

Let us see that $\mathbb{P}$ has reductants. In this particular case, the only head in the rules is A, so we have to define a reductant for A in $\mathbb{P}$.

From the two rules, we have the associated conjunctors to the implications $\&_P, \&_G$ (also defined componentwise), given its two truth-intervals $[\vartheta_1, \vartheta_2]$ and $[\tau_1, \tau_2]$ there should exist a truth-interval $[\epsilon_1, \epsilon_2]$, a conjunctor $\&$ and an aggregator $@$ such that for all $[b_1, b_2], [c_1, c_2] \in \mathcal{C}[0, 1]$:

$$\sup\{[\vartheta_1, \vartheta_2] \,\&_P [b_1, b_2], [\tau_1, \tau_2] \,\&_G [c_1, c_2]\} = [\epsilon_1, \epsilon_2] \,\&\, @([b_1, b_2], [c_1, c_2])$$

In this case, it suffices to consider $\& = \&_G$, $[\epsilon_1, \epsilon_2] = [\max(\vartheta_1, \tau_1), \min(\vartheta_2, \tau_2)]$, and $@([b_1, b_2], [c_1, c_2]) = [\max(\vartheta_1 b_1, c_1), \min(\vartheta_2 b_2, c_2)]$. □

Certainly, it will be interesting to consider only programs which contain all its reductants, but this might be a too heavy condition on our programs; the following proposition shows that it is not true, therefore we can assume that a program contains all its reductants, since its set of models is not modified.

Proposition 1. *Any reductant* $A \overset{\vartheta}{\leftarrow} \mathcal{B}$ *of* $\mathbb{P}$ *is satisfied by any model of* $\mathbb{P}$. *In short,* $\mathbb{P} \models A \overset{\vartheta}{\leftarrow} \mathcal{B}$.

It is possible *to construct reductants* for any head-of-rule in a given program under the only requirement that the truth-value set is complete under suprema; and this is actually an assumption for all multi-adjoint programs, which are based on a multi-adjoint lattice.

Definition 9 (Construction of reductants). *Let* $\mathbb{P}$ *be a multi-adjoint program; assume that all the rules in* $\mathbb{P}$ *with head* A *are* $\langle A \leftarrow_i \mathcal{B}_i, \vartheta_i \rangle$ *for* $i = 1, \ldots, n$. A reductant for A *is any rule* $\langle A \leftarrow @(\mathcal{B}_1, \ldots, \mathcal{B}_n), \top \rangle$ *where* $\leftarrow$ *is any implication with an adjoint conjunctor (let us denote it* $\&$*) and the aggregator* $@$ *is defined as follows*

$$\dot{@}(b_1, \ldots, b_n) = \sup\{\vartheta_1 \,\&_1\, b_1, \ldots, \vartheta_n \,\&_n\, b_n\}$$

It is immediate to prove that the rule constructed in the definition above is actually a reductant for A in $\mathbb{P}$, and we state the fact in the following proposition.

Proposition 2. *Under the hypotheses of the previous definition, the defined rule* $\langle A \leftarrow @(\mathcal{B}_1, \ldots, \mathcal{B}_n), \top \rangle$ *is a reductant for* A *in* $\mathbb{P}$.

Note that we have followed just traditional techniques of logic programming, and discarded non-determinism by using reductants. A possible disadvantage of this technique is that the full search space must be traversed (every rule of every atom must be evaluated), although this need not be necessary in many circumstances. It is clear that some evaluation strategies might start by executing non-deterministically the rules for a given atom, and finally the reductant. This joined with some memoizing or tabling technique would not have significant overhead, and could improve performance.

3.2 Completeness Results

The proof of the completeness theorems follows from some technical results. The first lemma below states that the least fix-point is also the least model of a program; the second states a characterisation of correct answers in terms of the $T_{\mathbb{P}}^{\mathfrak{L}}$ operator.

Lemma 1. *For all model I of $\mathbb{P}$ we have that $T_{\mathbb{P}}^{\omega}(\triangle) \sqsubseteq I$.*

Lemma 2. *$\lambda \in L$ is a correct answer for program $\mathbb{P}$ and query $?A$ if and only if $\lambda \preceq T_{\mathbb{P}}^{\omega}(\triangle)(A)$*

Now, in order to match correct and computed answers, the proposition below, whose proof is based induction on n, shows that any iteration of the $T_{\mathbb{P}}^{\mathfrak{L}}$ operator is, indeed, a computed answer.

Proposition 3. *Let $\mathbb{P}$ be a program, then $T_{\mathbb{P}}^{n}(\triangle)(A)$ is a computed answer for all n and for all query $?A$.*

We have now all the required background to prove a completeness result.

Theorem 2. *For every correct answer $\lambda \in L$ for a program $\mathbb{P}$ and a query $?A$, there exists a chain of elements λ_n such that $\lambda \preceq \sup \lambda_n$, such that for arbitrary n_0 there exists a computed answer δ such that $\lambda_{n_0} \preceq \delta$.*

Proof: Consider $\lambda_n = T_{\mathbb{P}}^{n}(\triangle)(A)$. As λ is a correct answer, we have that

$$\lambda \preceq T_{\mathbb{P}}^{\omega}(\triangle)(A) = \sup\{T_{\mathbb{P}}^{n}(\triangle)(A) \mid n \in \mathbb{N}\} = \sup \lambda_n$$

since $T_{\mathbb{P}}^{\omega}(\triangle)$ is a model. Now we can choose δ to be $T_{\mathbb{P}}^{n}(\triangle)(A)$ for any $n \geq n_0$ and the theorem follows directly by the monotonicity of the $T_{\mathbb{P}}^{\mathfrak{L}}$ operator and Proposition 3. □

The theorem above can be further refined under the assumption of the so-called supremum property:

Definition 10. *A cpo L is said to satisfy the* supremum property *if for all directed set $X \subset L$ and for all ε we have that if $\varepsilon < \sup X$ then there exists $\delta \in X$ such that $\varepsilon < \delta \leq \sup X$.*

Theorem 3 below states that any correct answer can be approximated up to any lower bound.

Theorem 3. *Assume L has the supremum property, then for every correct answer $\lambda \in L$ for a program $\mathbb{P}$ and a query $?A$, and arbitrary $\varepsilon \prec \lambda$ there exists a computed answer δ such that $\varepsilon \preceq \delta$.*

Proof: As λ is a correct answer, then $\lambda \preceq T_{\mathbb{P}}^{\omega}(\triangle)(A)$, since $T_{\mathbb{P}}^{\omega}(\triangle)$ is a model of $\mathbb{P}$. By definition we have $T_{\mathbb{P}}^{\omega}(\triangle)(A) = \sup\{T_{\mathbb{P}}^{n}(\triangle)(A) \mid n \in \mathbb{N}\}$ and, by hypothesis, $\varepsilon \prec \lambda \preceq T_{\mathbb{P}}^{\omega}(\triangle)(A)$. The supremum property states that there exists an element $\delta = T_{\mathbb{P}}^{n_0}(\triangle)(A)$ in $\{T_{\mathbb{P}}^{n}(\triangle)(A) \mid n \in \mathbb{N}\}$, such that $\varepsilon \prec \delta \preceq T_{\mathbb{P}}^{\omega}(\triangle)(A)$. This finishes the proof, for $T_{\mathbb{P}}^{n_0}(\triangle)(A)$ is a computed answer, by Proposition 3. □

4 Conclusions and Future Work

We have presented a framework for studying more elaborate proof procedures for multi-adjoint programs: a procedural semantics has been introduced, and two quasi-completeness theorems are stated and proved.

As future work, from the theoretical side, it is necessary to further investigate lattices with the supremum property; from the not-so-theoretical side, a practical evaluation of the proposed approach has to be performed, to evaluate the appropriateness of several possible optimisation techniques.

Acknowledgements

We thank C. Damásio and L. Moniz Pereira for their interesting comments on previous versions of this work. We are also grateful to the referees who pointed out several weaknesses of the first draft of the paper.

References

1. C.V. Damásio and L. Moniz Pereira. Monotonic and residuated logic programs. In *Sixth European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty, ECSQARU'01*, pages 748–759. Lect. Notes in Artificial Intelligence, 2143, Springer-Verlag, 2001.
2. M. Kifer and V. S. Subrahmanian. Theory of generalized annotated logic programming and its applications. *J. of Logic Programming*, 12:335–367, 1992.
3. J. Medina, M. Ojeda-Aciego, and P. Vojtáš. A completeness theorem for multi-adjoint logic programming. In *Proc. FUZZ-IEEE'01.* The 10th IEEE International Conference on Fuzzy Systems, IEEE Press, 2001. To appear.
4. J. Medina, M. Ojeda-Aciego, and P. Vojtáš. A multi-adjoint logic approach to abductive reasoning. In *Proc. 17th International Conference on Logic Programming, ICLP'01.* Lect. Notes in Artificial Intelligence 2273, Springer-Verlag, 2001.
5. J. Medina, M. Ojeda-Aciego, and P. Vojtáš. Multi-adjoint logic programming with continuous semantics. In *Proc. Logic Programming and Non-Monotonic Reasoning, LPNMR'01*, pages 351–364. Lect. Notes in Artificial Intelligence, 2173, Springer-Verlag, 2001.
6. E. Naito, J. Ozawa, I. Hayashi, and N. Wakami. A proposal of a fuzzy connective with learning function. In P. Bosc and J. Kaczprzyk, editors, *Fuzziness Database Management Systems*, pages 345–364. Physica Verlag, 1995.
7. P. Vojtáš and L. Paulík. Soundness and completeness of non-classical extended SLD-resolution. In *Proc. Extensions of Logic Programming*, pages 289–301. Lect. Notes in Comp. Sci. 1050, Springer-Verlag, 1996.

Representing and Reasoning on Three-Dimensional Qualitative Orientation Point Objects

Julio Pacheco Mª Teresa Escrig Francisco Toledo

Universitat Jaume I, Dpto. Ingeniería y Ciencia de los Computadores, Campus Riu Sec, 12071 Castellón (Spain)
e-mail: {pacheco | escrigm | toledo}@icc.uji.es

Abstract. An approach for representing and reasoning with 3-D qualitative orientation of point objects is presented. The model in 3-D is an extension of the Zimmerman and Freksa orientation model in 2-D. The paper presents attempts to represent 3-D spatial orientation in a final 3-D model. An iconic notation for 3-D spatial orientation relations is presented and the algebra is also explained.
Keywords: Spatial Reasoning, Qualitative Reasoning, Qualitative Orientation.

1 Introduction

One of the main aims of the Artificial Intelligence field is to simulate human behaviour in general and build robots with a human-like performance in particular. The principal goal of the Qualitative Spatial Reasoning field is to represent our everyday common sense knowledge about the physical world, and the underlying abstractions used by engineers and scientists when they create quantitative models. Kak [9] points out that the behaviour of the intelligent machine of the future might carry out temporal reasoning, spatial reasoning and also reason over interrelated entities occupying space and changing in time with respect to their attributes and spatial interrelationships. Spatial information that we obtain through perception is coarse and imprecise, thus qualitative models which reason with distinguishing characteristics rather than with exact measures seems to be more appropriate to deal with this kind of knowledge.

Supposing we want to know the qualitative orientation of the workmate's office in our university building (which has more than one floor) with respect the position we are. Or we know the relative orientation between some offices (in that building) and we want to know the orientation of every office with respect the rest of them. In that case, we need to know the height of every office, that is, we need to represent and reason with a 3-dimension orientation model.

There exist mainly three qualitative models for orientation which are not based on projections into external Reference Systems (RS): Freksa and Zimmermann's model [2, 3, 4, 5]; Hernández's approach [7, 8]; and Frank's approach [6]. In these models, space is divided into qualitative regions by means of RSs, which are centred on the reference objects. Spatial objects are always simplified to points, which are the representational primitives. From the three models not based on projections, the

Zimmerman and Freksa's model is considered more cognitive because no extrinsic reference system is necessary. This model has been chosen for extending to 3-D.

We are going to distinguish two parts in the reasoning process: the Basic Step of the Inference Process (BSIP) and the Full Inference Process (FIP). The BSIP can be defined in general terms such as: given a spatial relationship between point *c* with respect to (wrt) a RS, and another spatial relationship between point *d* wrt another RS, point *c* being part of that RS, the BSIP consists of obtaining the spatial relationship of point *d* wrt the first RS. When more relationships among several spatial landmarks are provided, then the FIP is necessary. It consists of repeating the BSIP as many times as possible, until no more information can be inferred.

To accomplish the integration of orientation, distance and cardinal directions into the same spatial model we will use the following three steps:

(1) the representation of the spatial aspect to be integrated;

(2) the definition of the BSIP for each represented spatial aspect; and

(3) the definition of the FIP for this spatial aspect.

In this paper, we are going to focus on those three parts.

2 The 2-D Zimmerman and Freksa Orientation Representation

In [2,3,4,5] approach, the orientation RS is defined by a point and a director vector *ab*, which defines the left/right dichotomy. The RS also includes the perpendicular line by the point *b*, which defines the first front/back dichotomy, and it can be seen as the straight line that joins our shoulders. This RS divides space into 9 qualitative regions. A finer distinction could be made in the back regions by drawing the perpendicular line by the point *a*. In this case, the space is divided into 15 qualitative regions. The point *a* defines the second front/back dichotomy of the RS.
The information represented is where the point *c* is with respect to the RS *ab*, that is, *c wrt ab*. This information can also be expressed as a result of applying the following operations: Identity, Inverse, Homing, Homing Inverse, Shortcut and ShortcutInverse.

3 The 3-D Orientation Model

3.1 Representation

We consider a plane that contain the two points which defines de 2-D RS, *a* and *b*.

We must decide on the plane before choosing the points [12]. In that case, our 3-D orientation RS will be based on a point and a reference plane. The reference plane chosen will be a plane parallel to the floor. In the case we do not have any specific plane to make reference, we must decide it first. When we said a reference plane we refer to *all the family of planes* parallel to the reference plane.

We consider two heights more that define the upper and downer height, respectively. Therefore, the 2-D orientation model has been extended to the third dimension, as it is shown in figure 1.

Considering these three heights we had extended the 15 qualitative regions in 45 qualitative regions.

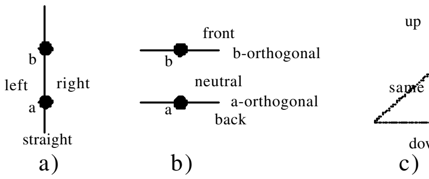


Fig. 1. a) left, straight and right; b) front, b-orthogonal, neutral, a-orthogonal and back; c) up, same and down

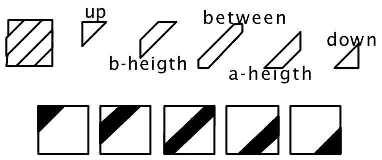


Fig. 2. A single cell divided into five heights and the names of every part; and the representation of the different heights.

Having these three heights (the ***up*** height, the ***same*** height and the ***down*** height with respect to the plane passes by the point *a*) implies that *a* and *b* have always to be in the same plane. In most of the cases this did not happen. As well as into the 2-D RS we can define a fine 3-D orientation RS by including five different heights: *up*, ***b-height***, *between*, ***a-height*** and *down*. (fig. 3 a). The 3-D orientation RS divides space into 75 qualitative regions. A 3-D iconic representation of the RS is shown in fig 3 b).

The names of every region are defined according to the position they are. We will use acronyms as `ulf` if it position is in *up-left-front*; `usf` for *up-straight-front*; `urf` for *up-right-front*, and so on (figure 4).

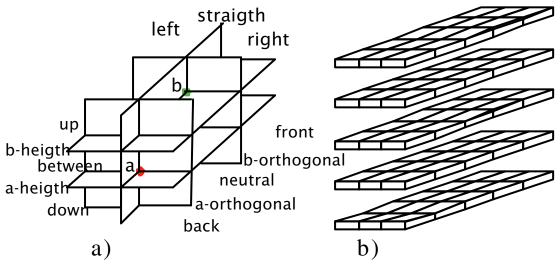


Fig. 3. a) the 3-D orientation RS, b) and the corresponding 3-D iconic representation.

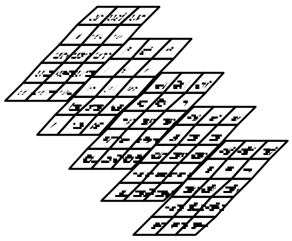


Fig. 4. the names inside of the 3-D iconic representation.

As a matter of clarity, the 3-D representation has been translated into 2-D iconic representation, as it is shown in figure 2. By agreement we will reason with the point *b* above the point *a*. For the cases in which the second point of the front/back dichotomy is not in the same plane or above the first point of the front/back dichotomy, we rotate the RS 180 degrees by using the spin operation.

The operations to be represented this 3-D orientation RS (*c wrt ab*) are: Identity, Spin, Inverse, Homing, Shortcut and its inverses. These will be defined in the algebra.

3.2 Algebra

The algebra consists on seven operations. (In [11,12] are represented the operations, its algebraic notation and its iconic representation).

The operations have been implemented as facts in a PROLOG database. In order to deal with the disjunction of relationships, the result of applying some operation to any orientation relationship is a list of relationships. Often this list contains only a relation (as in Identity, Inverse or Spin operations), but it allows us to deal with more than one relation if necessary (as in Homing, Homing Inverse, Shortcut and Shortcut Inverse).

- **Identity (id):** The PROLOG facts of the Identity operation would be for instance: `id(ulf,[ulf]);id(usf,[usf]);`etc.
- **Inversion (inv):** The PROLOG facts of the Inversion operation would be for instance: `inv(ulf,[drb]);inv(usf,[dsb]);`etc.
- **Spin (sp):** The PROLOG facts of the Spin operation would be for instance: `sp(ulf,[drf]);sp(usf,[dsf]);`etc.
- **Homing (hm):** The PROLOG facts of the Homing operation would be for instance: `hm(ulf,[dlb]);` `hm(usf,[dsb]);` etc. Here disjunction appear, for example: `hm(us,[dlf, dsf, drf, dl, ds, dr, dln, dsn, drn, dla, dsa, dra, dlb, dsb, drb])`.

When we complete the HM operation of the 3-D orientation relationship left-front-b-height (first row, second column), it happens that *c* and *b* are in the same plane. Therefore, we reduce the five heights to three (a-height, between and b-height are the same plane). In this case, the result of the HM operation is a disjunction because we have considered the two cases in which we applied or not the spin operation.

- **Homing Inverse (hmi):** The PROLOG facts of the Homing Inverse operation would be for instance: `hmi(ulf,[ulf]);` `hmi(usf,[usf]);` etc. Also disjunction appear here, for example: `hmi(us,[ulf, usf, urf, ul, us, ur, uln, usn, urn, ula, usa, ura, ulb, usb, urb]).`
- **Shortcut (sc):** The PROLOG facts of the Shortcut operation would be for instance: `sc(ulf,[brn]);sc(usf,[bsn]);`etc.
- **Shortcut Inverse (sci):** The PROLOG facts of the Shortcut Inverse operation would be for instance: `sci(ulf,[brn]);sci(usf,[bsn]);`etc.

3.3 Algebraic Combinations of Operations

There is a strong inner resemblance between the homing and the shortcut operations, for which only one table [12] is necessary, because all the results found in the homing 3-D iconic representation are found in the shortcut 3-D iconic representation.

We notice the resulting operation of combining two operations is other operation. E.g. INV(SC(x)) = SCI(x); and SC(INV(x)) = HM(x).

An important feature of these operations is their idempotent property [1]. The inverse, homing inverse and shortcut operations are idempotent of level two (i. e. INV(INV(*c wrt ab*)) = *c wrt ab)* and the homing and shortcut inverse operations are idempotent of level three. (i. e. HM(HM(HM(*c wrt ab*)))= *c wrt ab).*

Note that the application of the homing operation twice is equivalent to the application of the shortcut inverse operation once (i.e. SCI(HM(*c wrt ab*))= *c wrt ab*), and the application of the shortcut inverse operation twice is equivalent to apply the homing operation once (i.e. HM(SCI(*c wrt ab*))= *c wrt ab*).

4 The Basic Step of the Inference Process

The Basic Step of the Inference Process (BSIP) used in Freksa and Zimmermann's qualitative orientation approach is defined such as (figure 5): "given two relationships *c wrt ab* and *d wrt bc*, we want to obtain the relationship *d wrt ab*".

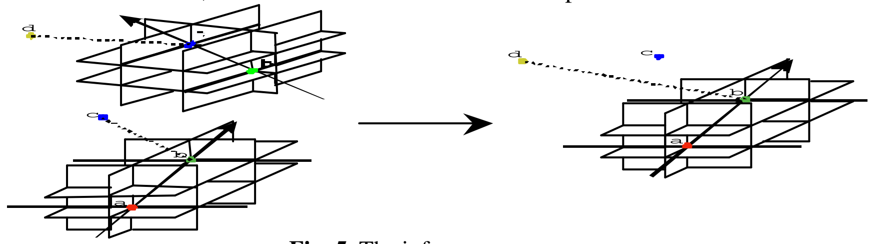


Fig. 5. The inference process.

The inference process among the coarse qualitative orientation relationships has been represented as an inference table of 75 x 75 entries using our approach [11]. The first column of the table shows the relationship *c wrt ab* and the first row of the table depicts the relationship *d wrt bc*. The relationship obtained in the composition table is *d wrt ab*. The result of the inference is always one of the seventy-five relationships or a disjunction of them.

4.1 The Inference Table

In order to visualise the reasoning procedure by means of tables, only a subset of the neighbourhood relationships is reflected. The arrangement of relationships in rows and columns of the fine 75 x 75 table are obtained following this idea (shown in [11]).

The PROLOG facts of the Inference Table would be for instance: `inf_table(ulf, ulf, [ulf, ul, uln, ula, ulb]);` `inf_table(usf, usf,[usf]);` etc.

5 The Full Inference Process

The Full Inference Process (FIP) is the other part in the reasoning process. When more relationships among several spatial landmarks are provided, then the FIP is necessary. Here a Constraint Solver (CS) for qualitative orientation will be explained. This CS, which implements a path consistency algorithm, is based on Constraint Logic Programming (CLP) extended with Constraint Handling Rules (CHRs).

5.1 Qualitative Orientation as a Constraint Satisfaction Problem

Our qualitative orientation model implies three spatial objects (*a*, *b* and *c*), therefore the constraints which deal with this information are tertiary. The Constrain Satisfaction Problem (CSP) is reformulated for these tertiary constraints (*c wrt ab*) such that: given a set of variables $\{X_1,...,X_n\}$, a discrete and finite domain for each variable {D1,...,Dn}, and a set of constraints $\{c_{c,ab}\ (X_c,X_a,X_b)\}$, which define the relationship between every group of three variables (X_c,X_a,X_b), $(1\leq a<b<c\leq n)$; the problem is to find an assignment of values (v_c,v_a,v_b), $v_i \in D_i$ to variables such that all constraints are satisfied, i.e. $c_{c,ab}$ (X_c,X_a,X_b) is true for every a,b,c $(1\leq a<b<c\leq n)$.

We redefine a network of tertiary constraints as path consistent for triples of nodes (c,a,b) and all paths a-b-i_1-...-i_{n-1}-i_n between them, if the direct constraint $c_{c,ab}$ is tighter (has less disjunction) than the indirect constraint along the path, i.e. the composition of constraints $c_{i1,ab} \otimes ... \otimes c_{c,i_{n-1}i_n}$ along the path. Until a fixed point is reached, to determine whether a graph is complete we repeat the following operation:

$$c_{d,ab} := c_{d,ab} \oplus c_{c,ab} \otimes c_{d,bc}$$

5.2 The Path Consistency Algorithm for Qualitative Orientation

The following constraint satisfaction algorithm for complete disjunctive tertiary constraints networks is defined using PROLOG extended with CHRs. PROLOG provides backtracking and CHRs are used to implement path consistency at a high level of abstraction.

The constraint $c_{c,ab}$ is represented in the algorithm by the predicate `cr_or(C,A,B,Rel)`, where Rel is the list of primitive spatial orientation relationships forming the disjunctive constraint. The operation of a path consistency, is implemented by means of two kinds of CHRs. The part of the operation corresponding to the intersection $c_{d,ab} \oplus ...$ is implemented by simplification CHRs:

```
cr_or(C,A,B,R1), cr_or(C,A,B,R2) <=>
          inter(R1,R2,R3) | cr_or(C,A,B,R3)
```

The part corresponding to the $c_{c,ab} \otimes c_{d,bc}$ is implemented by propagation CHRs:

```
cr_or(C,A,B,R1), cr_or(D,B,C,R2) ==>
          compo(R1,R2,R3) | newc(D,A,B,R3)
```

Termination is guaranteed because the simplification rule replaces R1 and R2 by the result R3 of intersecting R1 with R2 (and R3 is the same as R1 or R2 or smaller) and because propagation CHRs are never repeated for the same constraint goals as it will be shown.

The algorithm is based on the algorithm developed in [1]. The optimisation introduced in the algorithm of [10] (named PC-2) has also been included. This optimisation is based on the idea that the constraint $c_{c,ab}$ can be computed as the converse $c_{c,ab}$ if it is needed (by applying the inverse operation to the corresponding relationship), which saves half of the computation.

Here disjunction could also appear in first and second argument; in those cases operations of union, intersection and composition of disjunctive ternary constraints must be used.

The operations of *union*, *intersection* and *composition* are formally redefined for disjunctive ternary constraints in this section. (All these operations are associative).

The *union* of disjunctive ternary constraints can be formulated as follows:

$$c_{c,ab} \cup c'_{c,ab} := c\{r_1,..., r_n\}ab \ \vee c\{s_1,..., s_m\}ab = c\{r_1,..., r_n\} \cup c\{s_1,..., s_m\}ab$$

The *intersection* is defined as follows:

$$c_{c,ab} \cap \ c'_{c,ab} := c\{r_1,..., r_n\}ab \ \wedge c\{s_1,..., s_m\}ab = c\{r_1,..., r_n\} \cap \ c\{s_1,..., s_m\}ab$$

The *composition* of disjunctive ternary constraints is defined as follows:

$$c_{c,ab} \otimes \ c'd_{,ab} := c\{r_1,...,r_n\}ab \wedge d\{s_1,...,s_m\}bc = d\{r \otimes s | r \in \{r_1,..., r_n\}, s \in \{s_1,..., s_m\}\}ab$$

Where ⊗ is the basic step of inference process.

A disjunctive ternary constraint $c_{c,ab}$ between the variables a,b and c, also written $c\{r_1,..., r_n\}ab$, is a disjunction $(c\ r_1\ ab0\vee ...\vee (c\ r_n\ ab)$ where each r_i is a relation that is applicable to *c* and *ab*.

5.2.1 The Algorithm

Here a part of the path consistency algorithm to propagate compositions of disjunctive qualitative orientation relationships appear.

% Constraint declarations and definitions

```
constraints (cr_or)/4, (cr_or)/6.
label_with cr_or(N,C,A,B,Rel,I) if N>1.
cr_or(N,C,A,B,Rel,I) :- member(R,Rel), cr_or(1,C,A,B,[R],I).
```

% Initialise

```
cr_or(C,A,B,R)<=>length(R,N)|cr_or(N,C,A,B,R,1).
```

% Special cases

```
cr_or(N,C,A,B,R,I) <=> empty(R) | false.
cr_or(N,C,A,A,R,I) <=> true.
cr_or(N,C,C,B,R,I) <=> contains_equality_a(R) | true.
cr_or(N,C,A,C,R,I) <=> contains_equality_b(R) | true.
cr_or(N,C,A,B,R,I) <=> N=75 | true.
```

% Intersection

```
cr_or(N1,C,A,B,R1,I), cr_or(N2,C,A,B,R2,J) <=> inter(R1,R2,R3),
length(R3,N3), K is min(I,J) | cr_or(N3,C,A,B,R3,K).
cr_or(N1,B,C,A,R1,I), cr_or(N2,C,A,B,R2,J) <=> hm_op(R1,R11),
inter(R11,R2,R3),length(R3,N3),K is min(I,J)|cr_or(N3,C,A,B,R3,K).
cr_or(N1,C,A,B,R1,I), cr_or(N2,B,C,A,R2,J) <=> hm_op(R2,R22),
inter(R1,R22,R3),length(R3,N3),K is min(I,J)|cr_or(N3,C,A,B,R3,K)...
```

% Composition

```
cr_or(N1,C,A,B,R1,I), cr_or(N2,D,B,C,R2,J) ==> I=1,
comp(R1,R2,R3), length(R3,N3), K is I+J | cr_or(N3,D,A,B,R3,K).
cr_or(N1,B,C,A,R1,I),cr_or(N2,D,B,C,R2,J)==>I=1,singleton(R1),hm_op(
R1,R11),comp(R11,R2,R3),length(R3,N3),K is I+J|cr_or(N3,D,A,B,R3,K).
cr_or(N1,C,A,B,R1,I),cr_or(N2,C,D,B,R2,J)==>I=1,singleton(R2),hm_op(
R2,R22),comp(R1,R22,R3),length(R3,N3),K is I+J|cr_or(N3,D,A,B,R3,K)...
```

Predicates in **% Constraint declarations and definitions, %Initialize** and in **%Special cases** are introduced at the beginning of our database.

Simplification CHRs (rules in **% Intersection**) and propagation CHRs (rules in **% Composition**) perform intersections (which permit the simplification of redundant information) and compositions. The first rule implements (intersection or composition respectively) in the way such as it is originally defined in the rules in point 5.2.

By applying the five operations (HM, SC, INV, HMI and SCI) to the first constraint of the two which are in the head of the original rule (point 5.2), it is possible to obtain the orientation information among the same three spatial objects. (Only hm_op rule is exposed in this algorithm) Therefore, it is possible to calculate results if the corresponding operations are applied to the relationship or disjunction of relationships in the guard part of the rules. The application of the above operations to a disjunction of relationships is equivalent to the application of these operations to each relationship included in the disjunction of relations.

Notice that the operations that are idempotent of level three, have a different treatment to the rest of operations. If the HM operation is applied to the usual definition of the orientation ternary constraint (*c wrt ab*), the order of the variables in the constraint becomes (*a wrt bc*). However, if the HM operation is applied again to the result (to *a wrt bc*), instead of obtaining the original result, (which is the one expected when the operations are idempotent of level two), the relationship (*b wrt ca*) is achieved.

Second Simplification CHR corresponds to the above explanation third simplification CHR corresponds to the case in which the operations are applied to the second constraint of the two which appear in the head of the original intersection rule (the first rule in point 5.2). A total of 11 CHRs to compute results would be defined.

6 Conclusions and Future Works

We have left out of this paper some future work: the integration of different levels of granularity and the application of the 3-D orientation model to mobile robots with an arm manipulator on it.

Acknowledgements

This work has been supported by CICYT under grant number TAP99-0590-002-02 and by Fundació Caixa-Castelló BANCAIXA under grant number P1A98-11.

References

[1] Escrig, M.T., Toledo, F., Qualitative Spatial Reasoning: theory and practice. Application to Robot Navigation. IOS Press, Frontiers in A. I. and Applications, ISBN 90 5199 4125. 1998.

[2] Freksa, C., "Conceptual Neighbourhood and its role in temporal and spatial reasoning", Proceedings of IMACS Workshop on Decision Support Systems andQR, pg 181-187, 1991.

[3] Freksa, C., "Temporal reasoning based on semi-intervals", in Artificial Intelligence, vol. 54, pag. 199-227, 1992.

[4] Freksa, C., "Using Orientation Information for Qualitative Reasoning", in A. U. Frank, I. Campari, and U. Formentini (ed.). Theories and Methods of Spatio-Temporal Reasoning in Geographic Space. LNCS vol 639, Springer, Berlin, pag. 162-178, 1992.

[5] Freksa, C., Zimmermann, K., "On the Utilization of Spatial Structures for Cognitively Plausible and Efficient Reasoning", in Proceedings of the IEEE International Conference on Systems, Man and Cybernetics, pag. 18-21, 1992.

[6] Frank, A.U., "Qualitative Spatial Reasoning with cardinal directions", in Proceedings of the 7th Austrian Conference on Artificial Intelligence, Wien, Springer, Berlin, pg. 157-167,1991.

[7] Hernández, D., "Diagrammatical Aspects of Qualitative Representations of Space", Report FKI-164-92, Technische Universität München, Germany, 1992.

[8] Hernández, D., ``Qualitative Representation of Spatial Knowledge". In volume 804 of Lecture Notes in Artificial Intelligence. Ed. Springer-Verlag, 1994.

[9] Kak, A., "Spatial Reasoning", AI Magazine, vol.9, no. 2, p. 23, 1988.

[10] Mackworth, a.K., ``consistency in Networks of relations", Journal of Artificial Intelligence 8: pag. 99-118, 1977.

[11] Pacheco, J., Escrig, M.T., "A model for Representing and Reasoning with 3-D Qualitative Orientation" in the 4th Catalan congress of Artificial Intelligence, 2001.

[12] Pacheco, J., Escrig, M.T., Toledo, F. "An Approach to 3-D Qualitative Orientation of Point Objects" in the 9th Conference of Spanish Association of Artificial Intelligence, 2001.

Encodings for Equilibrium Logic and Logic Programs with Nested Expressions[⋆]

David Pearce[1], Hans Tompits[2], and Stefan Woltran[2]

[1] European Commission, DG Information Society – Future and Emerging Technologies
David.Pearce@cec.eu.int

[2] Institut für Informationssysteme, Abt. Wissensbasierte Systeme 184/3,
Technische Universität Wien, Favoritenstraße 9–11, A–1040 Vienna, Austria,
[tompits,stefan]@kr.tuwien.ac.at

Abstract. Equilibrium logic is an approach to nonmonotonic reasoning that generalises the stable model and answer set semantics for logic programs. We present a method to implement equilibrium logic and, as a special case, stable models for logic programs with nested expressions, based on polynomial reductions to quantified Boolean formulas (QBFs). Since there now exist efficient QBF-solvers, this reduction technique yields a practically relevant approach to rapid prototyping. The reductions for logic programs with nested expressions generalise previous results presented for other types of logic programs. We use these reductions to derive complexity results for the systems in question. In particular, we show that deciding whether a program with nested expressions has a stable model is Σ_2^P-complete.

1 Introduction

Equilibrium logic, developed in [22], is an approach to nonmonotonic reasoning that generalises the stable model and answer set semantics for logic programs. It is based on a nonclassical logic known as *here-and-there*, which is intermediate between classical logic and intuitionistic logic, and it provides a means to extend the reasoning mechanism associated with answer sets beyond the syntactic limitations of normal and disjunctive logic programs. Here we discuss a method to implement equilibrium logic and, as a special case, stable models for logic programs with nested expressions [19], based on polynomial reductions to quantified Boolean formulas (QBFs). A similar reduction technique was previously applied to other nonmonotonic reasoning formalisms, see, e.g., [7,5,3]. This in turn can be seen as a natural generalisation of a similar method successfully applied to NP-problems (see, e.g., [14]). Since there now exist efficient QBF-solvers [2,9,11,16,27], this reduction technique yields a practically relevant approach to rapid prototyping.

The reductions for logic programs with nested expressions generalise results presented for other types of logic programs in [7]. Although Lifschitz, Tang and Turner already provided a translation of nested expressions into an extension of standard logic programming [19], their translation is in general not polynomial because it is based on distributivity laws. In particular, this means that we cannot straightforwardly derive

[⋆] This work was partially supported by the Austrian Science Fund Project under grant P15068.

from it complexity results on reasoning with nested expressions. Here, however, we shall use our reduction techniques to discuss complexity issues as well as some other consequences of our results; in addition we deal briefly with some matters regarding implementation.

The remainder of the paper is laid out as follows. First we review the logical framework of quantified Boolean formulas. We then turn in Section 3 to equilibrium logic which is defined as a particular form of minimal-model reasoning in the logic of here-and-there. We show how to re-express satisfiability in here-and-there in the setting of classical logic and from there we proceed to characterise the central concept of equilibrium models using QBFs. Section 4 is devoted to logic programs with nested expressions as presented in [19]. Already [22] showed how stable models for normal and disjunctive programs can be regarded as equilibrium models. Later in [18] this result was extended to programs with nested expressions. From the main result of Section 3 it then follows that the property of being a stable model of this kind of program can also be expressed by QBFs. We show how to formulate this reduction in such a way that it leads to a linear translation of programs into QBFs.

Section 5 looks at the complexity of various reasoning tasks involving the logic of here-and-there and logic programs. In particular it is shown how the well-known property that deciding whether a disjunctive program has a stable model is Σ_2^P-complete extends to the case of nested expressions. Another result obtained here concerns the strong equivalence of logic programs, studied, e.g., in [18]. It was shown there that two programs are strongly equivalent if and only if they are logically equivalent in the logic of here-and-there. Our reduction techniques can be used to show that deciding strong equivalence of logic programs has coNP complexity.

In Section 6 we look briefly at an architecture for implementing the reductions and in Section 7 we draw conclusions and consider related and future work. The longer proof of Section 3 (Lemma 2) is relegated to an appendix.

2 Preliminaries

We deal with propositional languages and use the logical symbols $\top$, $\bot$, $\neg$, $\vee$, $\wedge$, and $\rightarrow$ to construct formulas in the standard way. We write $\mathcal{L}_{\mathcal{P}}$ to denote a language over an alphabet $\mathcal{P}$ of *propositional variables* or *atoms*. Formulas are denoted by Greek lower-case letters (possibly with subscripts). As usual, a *literal* is either an atom p (a *positive literal*) or a negated atom $\neg p$ (a *negative literal*). Furthermore, the logical complexity, $lc(\phi)$, of a formula ϕ is the number of occurrences of the logical symbols $\neg$, $\vee$, $\wedge$, and $\rightarrow$ in ϕ.

Given an alphabet $\mathcal{P}$, we define a disjoint alphabet $\mathcal{P}'$ as $\mathcal{P}' = \{p' \mid p \in \mathcal{P}\}$. Then, for $\alpha \in \mathcal{L}_{\mathcal{P}}$, we define α' as the result of replacing in α each atom p from $\mathcal{P}$ by the corresponding atom p' in $\mathcal{P}'$ (so implicitly there is an isomorphism between $\mathcal{P}$ and $\mathcal{P}'$). This is defined analogously for sets of formulas.

Quantified Boolean formulas (QBFs) generalise ordinary propositional formulas by the admission of quantifications over propositional variables (QBFs are denoted by Greek upper-case letters). Informally, a QBF of the form $\forall p\, \exists q\, \Phi$ means that for all truth assignments of p there is a truth assignment of q such that Φ is true. For instance, it is easily seen that the QBF $\exists p_1 \exists p_2 ((p_1 \to p_2) \wedge \forall p_3 (p_3 \to p_2))$ evaluates to true.

The precise semantical meaning of QBFs is defined as follows. First, some ancillary notation. An occurrence of a variable v in a QBF Φ is *free* iff it does not appear in the scope of a quantifier $\mathsf{Q}v$ ($\mathsf{Q} \in \{\forall, \exists\}$), otherwise the occurrence of v is *bound*. If Φ contains no free variables, then Φ is *closed*, otherwise Φ is *open*. Furthermore, $\Phi[v_1/\psi_1, \ldots, v_n/\psi_n]$ denotes the result of uniformly substituting the free occurrences of variables v_i in Φ by ψ_i ($1 \leq i \leq n$). By a (*classical*) *interpretation*, I, we understand a set of variables. Informally, a variable v is true under I iff $v \in I$. In general, the truth value, $\nu_I(\Phi)$, of a QBF Φ under an interpretation I is recursively defined as follows:

1. if $\Phi = \top$, then $\nu_I(\Phi) = 1$;
2. if $\Phi = \bot$, then $\nu_I(\Phi) = 0$;
3. if $\Phi = v$ is an atom, then $\nu_I(\Phi) = 1$ if $v \in I$, and $\nu_I(\Phi) = 0$ otherwise;
4. if $\Phi = \neg\Psi$, then $\nu_I(\Phi) = 1 - \nu_I(\Psi)$;
5. if $\Phi = (\Phi_1 \wedge \Phi_2)$, then $\nu_I(\Phi) = min(\{\nu_I(\Phi_1), \nu_I(\Phi_2)\})$;
6. if $\Phi = (\Phi_1 \vee \Phi_2)$, then $\nu_I(\Phi) = max(\{\nu_I(\Phi_1), \nu_I(\Phi_2)\})$;
7. if $\Phi = (\Phi_1 \to \Phi_2)$, then $\nu_I(\Phi) = 1$ iff $\nu_I(\Phi_1) \leq \nu_I(\Phi_2)$;
8. if $\Phi = \forall v \Psi$, then $\nu_I(\Phi) = \nu_I(\Psi[v/\top] \wedge \Psi[v/\bot])$;
9. if $\Phi = \exists v \Psi$, then $\nu_I(\Phi) = \nu_I(\Psi[v/\top] \vee \Psi[v/\bot])$.

We say that Φ is *true under I* iff $\nu_I(\Phi) = 1$, otherwise Φ is *false under I*. If $\nu_I(\Phi) = 1$, then I is a *model* of Φ. Likewise, for a set S of formulas, if $\nu_I(\Phi) = 1$ for all $\Phi \in S$, then I is a model of S. If Φ has some model, then Φ is said to be *satisfiable*. If Φ is true under any interpretation, then Φ is *valid*. Observe that a closed QBF is either valid or unsatisfiable, because closed QBFs are either true under each interpretation or false under each interpretation. Hence, for closed QBFs, there is no need to refer to particular interpretations.

We note the following elementary property, which will be relevant later on. Let Φ be a QBF whose free variables are given by V, and let $W \subseteq V$. Then, I is a model of $\exists W \Phi$ iff there is some $J \subseteq W$ such that $J \cup I$ is a model of Φ.

In the sequel, we employ the following abbreviations in the context of QBFs: Let $S = \{\phi_1, \ldots, \phi_n\}$ and $T = \{\psi_1, \ldots, \psi_n\}$ be indexed sets of formulas. Then, $S \leq T$ abbreviates $\bigwedge_{i=1}^{n}(\phi_i \to \psi_i)$, and $S < T$ is a shorthand for $(S \leq T) \wedge \neg(T \leq S)$. Furthermore, for a set $P = \{p_1, \ldots, p_n\}$ of propositional variables and a quantifier $\mathsf{Q} \in \{\forall, \exists\}$, we let $\mathsf{Q}P\,\Phi$ stand for the formula $\mathsf{Q}p_1\mathsf{Q}p_2 \cdots \mathsf{Q}p_n\,\Phi$. Finally, finite sets $T = \{\phi_1, \ldots, \phi_n\}$ of formulas are usually identified with the conjunction $\bigwedge_{i=1}^{n} \phi_i$ of their elements.

The operators $\leq$ and $<$ are fundamental tools for expressing certain tests on sets of atoms. In particular, the following properties hold: Let $V = \{v_1, \ldots, v_n\}$ and $W = \{w_1, \ldots, w_n\}$ be two sets of indexed atoms, and let $I_1 \subseteq V$ and $I_2 \subseteq W$ be two interpretations. Then, (i) $I_1 \cup I_2'$ is a model of $V \leq W'$ iff $I_1 \subseteq I_2$, and (ii) $I_1 \cup I_2'$ is a model of $V < W'$ iff $I_1 \subset I_2$.

3 Equilibrium Logic

We start our discussion with equilibrium logic, an approach to nonmonotonic reasoning that generalises the answer set semantics [10] for logic programs. Equilibrium logic was introduced in [22] and further investigated in [23]; proof theoretic studies of the logic can be found in [25,24]. The main difference here is that we do not consider languages with a second kind of negation, strong negation, needed for capturing answer set semantics for logic programs with two kinds of negation.

The main result of this section is to show how equilibrium logic can be mapped to quantified Boolean formulas in polynomial time. This translation will be constructed in two steps: First, since equilibrium logic is defined as a special form of minimal-model reasoning in the (nonclassical) logic of here-and-there, we translate the logic of here-and-there into classical logic. Afterwards, we use this translation to encode equilibrium logic in terms of QBFs.[1]

Generally speaking, the logic of here-and-there is an important tool for analysing various properties of logic programs. For instance, as shown in [18], the problem of checking whether two logic programs are strongly equivalent can be expressed in terms of the logic of here-and-there.

In what follows, we use the propositional language $\mathcal{L}_{\mathcal{P}}$ defined in Section 2, where $\mathcal{P}$ is some alphabet.

The semantics of the logic of here-and-there is defined in terms of two worlds, H and T, called "here" and "there". It is assumed that there is a total order, $\leq$, defined between these worlds such that $\leq$ is reflexive and $H \leq T$. As in ordinary Kripke semantics for intuitionistic logic, we can imagine that in each world a set of atoms is verified and that, once verified "here", an atom remains verified "there". Formally, by an *HT-interpretation*, $\mathcal{I}$, we understand an ordered pair $\langle I_H, I_T \rangle$ of sets of atoms such that $I_H \subseteq I_T$. Then, the truth value, $\nu_{\mathcal{I}}(w, \phi)$, of a formula ϕ at a world $w \in \{H, T\}$ in an HT-interpretation $\mathcal{I} = \langle I_H, I_T \rangle$ is recursively defined as follows:

1. if $\phi = \top$, then $\nu_{\mathcal{I}}(w, \phi) = 1$;
2. if $\phi = \bot$, then $\nu_{\mathcal{I}}(w, \phi) = 0$;
3. if $\phi = v$ is an atom, then $\nu_{\mathcal{I}}(w, \phi) = 1$ if $v \in I_w$, and $\nu_{\mathcal{I}}(w, \phi) = 0$ otherwise;
4. if $\phi = \neg\psi$, then $\nu_{\mathcal{I}}(w, \phi) = 1$ if for every world u such that $w \leq u$, $\nu_{\mathcal{I}}(u, \psi) = 0$, and $\nu_{\mathcal{I}}(w, \phi) = 0$ otherwise;
5. if $\phi = (\phi_1 \wedge \phi_2)$, then $\nu_{\mathcal{I}}(w, \phi) = min(\{\nu_{\mathcal{I}}(w, \phi_1), \nu_{\mathcal{I}}(w, \phi_2)\})$;
6. if $\phi = (\phi_1 \vee \phi_2)$, then $\nu_{\mathcal{I}}(w, \phi) = max(\{\nu_{\mathcal{I}}(w, \phi_1), \nu_{\mathcal{I}}(w, \phi_2)\})$;
7. if $\phi = (\phi_1 \rightarrow \phi_2)$, then $\nu_{\mathcal{I}}(w, \phi) = 1$ if for every world u such that $w \leq u$, $\nu_{\mathcal{I}}(u, \phi_1) \leq \nu_{\mathcal{I}}(u, \phi_2)$, and $\nu_{\mathcal{I}}(w, \phi) = 0$ otherwise.

We say that ϕ is *true under* $\mathcal{I}$ *in* w iff $\nu_{\mathcal{I}}(w, \phi) = 1$, otherwise ϕ is *false under* $\mathcal{I}$ *in* w. An HT-interpretation $\mathcal{I} = \langle I_H, I_T \rangle$ *satisfies* ϕ, or $\mathcal{I}$ is an *HT-model* of ϕ, iff $\nu_{\mathcal{I}}(H, \phi) = 1$. If ϕ is true under any HT-interpretation, then ϕ is *HT-valid*. An HT-interpretation $\langle I_H, I_T \rangle$ is *total* if $I_H = I_T$.

It is easily seen that any HT-valid formula is valid in classical logic, but the converse does not always hold. For instance, $p \vee \neg p$ and $\neg\neg p \rightarrow p$ are valid in classical logic but not in the logic of here-and-there, because $\mathcal{I} = \langle \emptyset, \{p\} \rangle$ is not an HT-model for either of these formulas.

[1] The logic of here-and-there, whose name is motivated below, is also commonly known as Gödel's 3-valued logic in view of Gödel's paper [12]. It was first presented in the form of truth matrices by Heyting in [13] and first axiomatised by Lukasiewicz in [21].

Equilibrium logic can be seen as a particular type of reasoning with minimal HT-models. Formally, an *equilibrium model* of a formula ϕ is a total HT-interpretation $\langle I, I\rangle$ such that (i) $\langle I, I\rangle$ is an HT-model of ϕ, and (ii) for every proper subset J of I, $\langle J, I\rangle$ is not an HT-model of ϕ.

We proceed with the following two results, which we use frequently later on:

Proposition 1. *For any HT-interpretation $\mathcal{I} = \langle I_H, I_T\rangle$ and any propositional formula ϕ, the following relations hold:*

1. $\nu_{\mathcal{I}}(T, \phi) = 1$ *iff* $\nu_{I_T}(\phi) = 1$*;*
2. $\nu_{\mathcal{I}}(H, \phi) = 1$ *implies* $\nu_{\mathcal{I}}(T, \phi) = 1$.

Observe that the first part of this proposition states that ϕ is true under $\mathcal{I} = \langle I_H, I_T\rangle$ in the world T iff ϕ is true under I_T in classical logic.

Proposition 2. *A total HT-interpretation $\langle I, I\rangle$ is an HT-model of ϕ iff I is a model of ϕ in classical logic.*

Next, we express satisfiability in the logic of here-and-there in terms of satisfiability in classical logic. We start with the following translation:

Definition 1. *Let ϕ be a formula. Then, $\tau[\phi]$ is recursively defined as follows:*

1. *if ϕ is an atom, or one of $\top$ or $\bot$, then $\tau[\phi] = \phi$;*
2. *if $\phi = (\phi_1 \circ \phi_2)$, for $\circ \in \{\wedge, \vee\}$, then $\tau[\phi] = \tau[\phi_1] \circ \tau[\phi_2]$;*
3. *if $\phi = \neg\psi$, then $\tau[\phi] = \neg\tau[\psi] \wedge \neg\psi'$;*
4. *if $\phi = (\phi_1 \rightarrow \phi_2)$, then $\tau[\phi] = (\tau[\phi_1] \rightarrow \tau[\phi_2]) \wedge (\phi_1' \rightarrow \phi_2')$.*

First of all, let us note that this transformation is quadratic in the size of the input formula. More precisely, we have the following property:

Lemma 1. *Let $\tau[\cdot]$ be the transformation defined above. Then,*

$$lc(\tau[\phi]) \leq lc(\phi)(lc(\phi) + 2), \textit{ for any formula } \phi.$$

Proof. By induction on $lc(\phi)$. □

Intuitively, the primed formulas in $\tau[\phi]$ correspond to formulas evaluated in the world "there", whilst unprimed formulas correspond to formulas evaluated in "here". In order to fully express the semantics of the logic of here-and-there, the only additional requirement needed is to ensure that all formulas true in "here" are also true in "there". This can be conveniently expressed in terms of the condition $V \leq V'$, where V is the set of atoms occurring in ϕ. More formally, we have the following relation:

Lemma 2. *Let ϕ be a formula on atoms V and let $I_H, I_T \subseteq V$ be interpretations. Then, $\langle I_H, I_T\rangle$ is an HT-model of ϕ iff $I_H \cup I_T'$ is a model of*

$$\mathcal{T}_{HT}[\phi] = (V \leq V') \wedge \tau[\phi].$$

Proof. See Appendix A. □

Equilibrium models are then expressed as follows:

Theorem 1. *Let ϕ be a formula on atoms V. Then, $\langle I, I\rangle$ is an equilibrium model of ϕ iff I' is a model of*

$$\mathcal{T}_E[\phi] = \phi' \wedge \neg\exists V((V < V') \wedge \tau[\phi]).$$

Proof. We first note that $\mathcal{T}_E[\phi]$ is obviously equivalent to

$$\phi' \wedge \neg\exists V((V < V') \wedge \mathcal{T}_{HT}[\phi]), \tag{1}$$

because $(V < V')$ is logically equivalent to $(V < V') \wedge (V \leq V')$.

Let $I \subseteq V$ be some interpretation. Recall that $\langle I, I\rangle$ is an equilibrium model of ϕ iff (i) $\langle I, I\rangle$ is an HT-model of ϕ, and (ii) for every $J \subset I$, $\langle J, I\rangle$ is not an HT-model of ϕ. We show that (i) and (ii) hold iff I' is a model of (1).

First of all, by Proposition 2 and a simple renaming, it follows that Condition (i) holds iff ϕ' is true under I'. Now consider the QBF $\Psi = \exists V((V < V') \wedge \mathcal{T}_{HT}[\phi])$. By the properties of $<$ and the semantics of the existential quantifier, we have that Ψ is true under I' iff there is some $J \subset I$ such that $\mathcal{T}_{HT}[\phi]$ is true under $J \cup I'$. Hence, invoking Lemma 2, we get that Ψ is true under I' iff Condition (ii) does not hold. Consequently, (ii) holds iff the second conjunct of (1) is true under I'. Therefore, (i) and (ii) are jointly satisfied iff I' is a model of (1). □

Observe that, in virtue of Lemma 1, both encodings $\mathcal{T}_{HT}[\cdot]$ and $\mathcal{T}_E[\cdot]$ are computable in polynomial time with respect to the size of the input formula.

To illustrate the mechanism of both transformations, consider the formula $\phi = \neg\neg p \rightarrow p$. We already pointed out that ϕ is not valid in the logic of here-and-there, although it is clearly a tautology of classical propositional logic. Let us first construct $\tau[\phi]$, which is given by

$$(\tau[\neg\neg p] \rightarrow \tau[p]) \wedge (\neg\neg p' \rightarrow p').$$

Applying the definition of $\tau[\cdot]$ recursively, we get

$$\left[\Big(\neg(\neg p \wedge \neg p') \wedge \neg\neg p'\Big) \rightarrow p\right] \wedge (\neg\neg p' \rightarrow p'). \tag{2}$$

The first conjunct of (2) is equivalent to $p' \rightarrow p$, and the second conjunct is a tautology. Hence, $\mathcal{T}_{HT}[\phi] = (p \rightarrow p') \wedge \tau[\phi]$ is equivalent to

$$(p \rightarrow p') \wedge (p' \rightarrow p),$$

which has two models, viz. $M_1 = \{\}$ and $M_2 = \{p, p'\}$. Therefore, by Lemma 2, the HT-models of $\neg\neg p \rightarrow p$ are given by $\langle\emptyset, \emptyset\rangle$ and $\langle\{p\}, \{p\}\rangle$. Observe that $\langle\emptyset, \{p\}\rangle$ is not an HT-model of ϕ, which is in accordance with our discussion above.

Concerning the transformation $\mathcal{T}_E[\phi]$, let us compute the equilibrium models of $\neg\neg p \rightarrow p$ by means of Theorem 1. Since $\tau[\phi]$ is equivalent to $p' \rightarrow p$, we get that $\mathcal{T}_E[\phi]$ is equivalent to

$$(\neg\neg p' \rightarrow p') \wedge \neg\exists p\Big((p \rightarrow p') \wedge \neg(p' \rightarrow p) \wedge (p' \rightarrow p)\Big). \tag{3}$$

Observe that $\neg(p' \rightarrow p) \wedge (p' \rightarrow p)$ makes the whole formula in the scope of $\exists p$ unsatisfiable, so (3) is equivalent to $(\neg\neg p' \rightarrow p')$, which is a tautology of classical logic. Thus, the models of $\mathcal{T}_E[\phi]$ are given by $\emptyset$ and p', and Theorem 1 implies that the equilibrium models of $(\neg\neg p' \rightarrow p')$ are given by $\langle\emptyset, \emptyset\rangle$ and $\langle\{p\}, \{p\}\rangle$.

4 Logic Programs

In this section we discuss logic programs with nested expressions, following Lifschitz *et al.* [19].[2] These kinds of programs generalise normal logic programs by allowing bodies and heads of rules to contain arbitrary Boolean formulas. We show that logic programs with nested expressions can be encoded in terms of QBFs in linear time, exploiting the reduction of equilibrium logic discussed in the previous section. Based on the resulting transformations, in Section 5 we derive several complexity results related to logic programs with nested expressions.

We start with some basic notation. A formula of $\mathcal{L}_\mathcal{P}$ whose sentential connectives comprise only $\wedge$, $\vee$, or $\neg$ is called an *expression.* A rule, r, is an ordered pair of the form

$$H(r) \leftarrow B(r),$$

where $B(r)$ and $H(r)$ are expressions. We call $B(r)$ the *body* of r and $H(r)$ the *head* of r. A *program*, Π, is a finite set of rules.

We employ for rules and programs the same notational convention concerning priming as we did for formulas, i.e., r' is the result of replacing each atom v in r by v' and, similarly, Π' is the result of replacing each $r \in \Pi$ by r'.

Note that programs properly generalise *disjunctive logic programs* [10], which are characterised by the condition that bodies of rules are conjunctions of literals, and heads are disjunctions of atoms.

In what follows, we associate to each rule r a corresponding formula $\hat{r} = B(r) \rightarrow H(r)$ and, accordingly, to each program Π a corresponding set of formulas $\hat{\Pi} = \{\hat{r} \mid r \in \Pi\}$.

We call expressions, rules, and programs *basic* iff they do not contain the operator $\neg$. An interpretation I is a *model* of a basic program Π if it is a model of the associated set $\hat{\Pi}$ of formulas.

Given an interpretation I and an (arbitrary) program Π, the *reduct*, Π^I, of Π with respect to I is the basic program obtained from Π by replacing every occurrence of an expression $\neg\psi$ in Π which is not in the scope of any other negation by $\bot$ if ψ is true under I, and by $\top$ otherwise. I is a *stable model* of Π iff it is a minimal model (with respect to set inclusion) of the reduct Π^I.

A formula ϕ is said to be a *brave consequence* of a logic program Π iff there is a stable model I of Π such that ϕ is true under I, and ϕ is a *skeptical consequence* of Π iff ϕ is true under all stable models I of Π.

Two logic programs, Π_1 and Π_2, are *equivalent* iff they possess the same stable models. Following Lifschitz *et al.* [18], we call Π_1 and Π_2 *strongly equivalent* iff, for

[2] Here we consider languages with only one kind of negation, however, corresponding to default negation.

every program Π, $\Pi_1 \cup \Pi$ and $\Pi_2 \cup \Pi$ are equivalent. The following property was shown in [18]:

Proposition 3. *Let Π_1 and Π_2 be programs, and let $\hat{\Pi}_i = \{B(r) \to H(r) \mid r \in \Pi_i\}$, for $i = 1, 2$. Then, Π_1 and Π_2 are strongly equivalent iff $\hat{\Pi}_1$ and $\hat{\Pi}_2$ are equivalent in the logic of here-and-there.*

Strong equivalence is a useful concept to simplify parts of programs. In Section 5, we analyse the computational cost of deciding whether two programs are strongly equivalent.

The following result establishes the close connection between equilibrium models and stable models, showing that stable models are actually a special case of equilibrium models:

Proposition 4 ([18]). *For any program Π, I is a stable model of Π iff $\langle I, I\rangle$ is an equilibrium model of $\hat{\Pi}$.*

In concluding our review of logic programs, let us note that in [19] a transformation is presented which maps logic programs with nested expressions into equivalent disjunctive logic programs.[3] However, this translation is in general not polynomial because it relies on distributive laws which yield an exponential explosion in some instances. Indeed, this is, e.g., always the case whenever the head of a rule is an expression in disjunctive normal form, or the body is an expression in conjunctive normal form. In this section, on the other hand, we shall construct a translation of programs into QBFs which is *linear* in the size of the input program.

The first step towards this translation is the following QBF encoding of logic programs, which is an immediate consequence of Theorem 1 and Proposition 4:

Theorem 2. *Let Π be a logic program, $\hat{\Pi} = \{B(r) \to H(r) \mid r \in \Pi\}$ the set of formulas associated with Π, and V the set of atoms occurring in Π. Then, $I \subseteq V$ is a stable model of Π iff I' is a model of*

$$\mathcal{T}_E[\hat{\Pi}] = \hat{\Pi}' \wedge \neg\exists V((V < V') \wedge \tau[\hat{\Pi}]).$$

As we know from Section 3, this encoding is quadratic in the size of the input program. In order to get a linear transformation, we consider the following adaption of transformation $\tau[\cdot]$:

Definition 2. *Let ϕ be an expression. Then, $\tau^*[\phi]$ is recursively defined as follows:*

1. *if ϕ is an atom, or one of $\top$ or $\bot$, then $\tau^*[\phi] = \phi$;*
2. *if $\phi = (\phi_1 \circ \phi_2)$, for $\circ \in \{\wedge, \vee\}$, then $\tau^*[\phi] = \tau^*[\phi_1] \circ \tau^*[\phi_2]$;*
3. *if $\phi = \neg\psi$, then $\tau^*[\phi] = \neg\psi'$.*

Obviously $lc(\tau^*[\phi]) = lc(\phi)$, for any expression ϕ. Observe that the difference between $\tau^*[\phi]$ and $\tau[\phi]$ lies in the treatment of negation: in $\tau^*[\phi]$ the recursion comes to a halt if ϕ is a negated formula. Intuitively, this follows from an application of Part 2 of Proposition 1 for negated formulas.

[3] Observe that [19] allows literals to occur in heads as well as in bodies.

Lemma 3. *Let ϕ be an expression and $\langle I_H, I_T\rangle$ an HT-interpretation. Then, $I = I_H \cup I'_T$ is a model of $\tau[\phi]$ iff it is a model of $\tau^*[\phi]$.*

Proof. By induction on $lc(\phi)$.

INDUCTION BASE. Assume $lc(\phi) = 0$. Then, $\tau[\phi] = \tau^*[\phi]$ and the statement holds trivially.

INDUCTION STEP. Assume $lc(\phi) > 0$, and let the statement hold for all expressions ψ such that $lc(\psi) < lc(\phi)$. Suppose $\phi = (\phi_1 \circ \phi_2)$, for $\circ \in \{\wedge, \vee\}$. Then, $\tau[\phi] = \tau[\phi_1] \circ \tau[\phi_2]$ and $\tau^*[\phi] = \tau^*[\phi_1] \circ \tau^*[\phi_2]$. Since $lc(\phi_i) < lc(\phi)$, for $i = 1, 2$, by induction hypothesis we get that $\tau[\phi_1]$ and $\tau^*[\phi_1]$, as well as $\tau[\phi_2]$ and $\tau^*[\phi_2]$, have the same models. Consequently, $\tau[\phi]$ and $\tau^*[\phi]$ possess identical models.

Now assume that $\phi = \neg\psi$, for some expression ψ. Then, $\tau^*[\phi] = \neg\psi'$. Since $\tau[\phi] = \neg\tau[\psi] \wedge \tau^*[\phi]$, each model of $\tau[\phi]$ is also a model of $\tau^*[\phi]$. So suppose that $I = I_H \cup I'_T$ is a model of $\tau^*[\phi] = \neg\psi'$, where $\mathcal{I} = \langle I_H, I_T\rangle$ is an HT-interpretation. Now, I is a model of $\neg\psi'$ iff $\nu_{I'_T}(\neg\psi') = 1$. Hence, renaming yields $\nu_{I_T}(\neg\psi) = 1$. By Part 1 of Proposition 1, we get that $\nu_{\mathcal{I}}(T, \psi) = 0$, which in turns implies $\nu_{\mathcal{I}}(H, \psi) = 0$, in virtue of Part 2 of Proposition 1. Hence, $\mathcal{I}$ is an HT-model of $\neg\psi$. Applying Lemma 2, we conclude that $I_H \cup I'_T$ is a model of $\tau[\neg\psi]$. □

Given this result, the optimised QBF encoding for logic programs is as follows:

Theorem 3. *Let Π, $\hat{\Pi}$, and V be as in Theorem 2. Then, $I \subseteq V$ is a stable model of Π iff I' is a model of*

$$\mathcal{T}_S[\Pi] = \hat{\Pi}' \wedge \neg\exists V\Big((V < V') \wedge \bigwedge_{r\in\Pi} \big(\tau^*[B(r)] \rightarrow \tau^*[H(r)]\big)\Big).$$

Proof. We show that $\mathcal{T}_S[\Pi]$ is equivalent to

$$\mathcal{T}_E[\hat{\Pi}] = \hat{\Pi}' \wedge \neg\exists V\big((V < V') \wedge \tau[\hat{\Pi}]\big).$$

First of all, by definition of $\tau[\cdot]$, we get

$$\tau[\hat{\Pi}] = \bigwedge_{r\in\Pi} (\tau[B(r)] \rightarrow \tau[H(r)]) \wedge \hat{\Pi}'.$$

Now consider the formula $(V < V') \wedge \tau[\hat{\Pi}]$. Since $(V < V') = (V \leq V') \wedge \neg(V' \leq V)$, any model of $(V < V') \wedge \tau[\hat{\Pi}]$ is also a model of $(V \leq V')$. Furthermore, recall that for any interpretation $J_1 \cup J'_2$ with $J_i \subseteq V$ $(i = 1, 2)$, we have that $\langle J_1, J_2\rangle$ is an HT-interpretation iff $J_1 \cup J'_2$ is a model of $V \leq V'$. Hence, by applying Lemma 3, it follows that $(V < V') \wedge \tau[\hat{\Pi}]$ is equivalent to

$$(V < V') \wedge \bigwedge_{r\in\Pi} (\tau^*[B(r)] \rightarrow \tau^*[H(r)]) \wedge \hat{\Pi}'.$$

We obtain that $\mathcal{T}_E[\hat{\Pi}]$ is equivalent to

$$\hat{\Pi}' \wedge \neg\exists V\Big((V < V') \wedge \bigwedge_{r\in\Pi} (\tau^*[B(r)] \rightarrow \tau^*[H(r)]) \wedge \hat{\Pi}'\Big). \quad (4)$$

Clearly, the occurrence of $\hat{\Pi}'$ in the scope of $\exists V$ can be moved outside the quantifier, which gets then absorbed by the first conjunct of (4). The result of this manipulation is $\mathcal{T}_S[\Pi]$. □

The above transformation also allows us to express the basic consequence relations associated with logic programs in terms of QBFs, as stated next:

Corollary 1. *Let Π be a logic program on atoms V, let ϕ be a formula, and let U be the set of atoms occurring in ϕ but not in Π. Then,*

1. *ϕ is a brave consequence of Π iff $\exists V'(\mathcal{T}_S[\Pi] \wedge \forall U'\phi')$ is true; and*
2. *ϕ is a skeptical consequence of Π iff $\forall V'(\mathcal{T}_S[\Pi] \rightarrow \forall U'\phi')$ is true.*

Theorems 2 and 3 generalise a similar QBF encoding given in [7] for the case of disjunctive logic programs. There, the following QBF was used to evaluate a given disjunctive logic program Π:

$$\mathcal{T}_{DLP}[\Pi] = \hat{\Pi} \wedge \neg\exists V'\Big[(V' < V) \wedge \bigwedge_{r\in\Pi} \Big(B^+(r') \wedge B^-(r)) \rightarrow H(r')\Big)\Big].$$

In this formula, $B^+(r)$ denotes the conjunction of all positive literals in $B(r)$, and $B^-(r)$ denotes the conjunction of all negative literals in $B(r)$.

It is easily seen that, given a disjunctive logic program Π, the transformation $\mathcal{T}_S[\Pi]$ coincides with $\mathcal{T}_{DLP}[\Pi]$, providing the priming of formulas is interchanged.

5 Complexity

In this section, we analyse the complexity of several decision problems associated with the logic of here-and-there and logic programs, respectively. Besides the reasoning tasks underlying the QBF encodings from the previous sections, we also deal with the complexity of determining whether two logic programs are strongly equivalent.

For each of the tasks discussed in the following, upper complexity bounds can be obtained by the quantifier structure of the corresponding QBF encoding. In fact, the basis for this is the following well-known result.

Proposition 5. *Let ϕ be propositional formula whose atoms are partitioned into $i \geq 1$ sets $V_1, \ldots, V_i$, and let $\Phi = \mathsf{Q}_1 V_1 \ldots \mathsf{Q}_i V_i\, \phi$ be a QBF in prenex form, where $\mathsf{Q}_j \in \{\exists, \forall\}$ and $\mathsf{Q}_k \neq \mathsf{Q}_{k+1}$, for $1 \leq j \leq i$ and $1 \leq k < i$. Then, deciding whether Φ evaluates to true is*

(a) *Σ_i^P-complete if $\mathsf{Q}_1 = \exists$, and*
(b) *Π_i^P-complete if $\mathsf{Q}_1 = \forall$.*

Recall that Σ_i^P and Π_i^P are the constituting members of the polynomial hierarchy, where $\Sigma_0^P = \Pi_0^P = \mathrm{P}$, $\Sigma_1^P = \mathrm{NP}$, and $\Pi_1^P = \mathrm{coNP}$.

We first deal with the satisfiability problem of the logic of here-and-there.

Theorem 4. *Deciding whether a propositional formula has an HT-model is* NP*-complete.*

Proof. Membership follows from the polynomial-time constructible reduction $\mathcal{T}_{HT}[\cdot]$ into classical propositional logic. Hardness can be shown by reducing the satisfiability problem for a classical propositional formula in conjunctive normal form into the satisfiability problem in the logic of here-and-there. This can be done as follows.

Let ϕ be a formula in conjunctive normal form, let $V = \{v_1, \ldots, v_n\}$ be the set of atoms occurring in ϕ, and let $W = \{w_1, \ldots, w_n\}$ be a set of new atoms. Then, ϕ is satisfiable in classical propositional logic iff

$$\bigwedge_{i=1}^{n} \Big((v_i \vee w_i) \wedge (\neg v_i \vee \neg w_i) \Big) \wedge \tilde{\phi},$$

has an HT-model, where $\tilde{\phi}$ results from ϕ by replacing each negative literal $\neg v_i$ in ϕ by w_i. □

Next, we turn our attention to the basic reasoning tasks associated with logic programs. The following proposition was shown in [8]:

Proposition 6. *Deciding whether a disjunctive logic program has at least one stable model is Σ_2^P-complete.*

We extend this result as follows:

Theorem 5. *Deciding whether a logic program* (*containing nested expressions*) *has at least one stable model is Σ_2^P-complete.*

Proof. Theorem 3 implies that a program Π has a stable model iff $\exists V' \mathcal{T}_S[\Pi]$ evaluates to true, where V is the set of atoms occurring in Π. Hence, membership follows immediately from Proposition 5 by observing that $\exists V' \mathcal{T}_S[\Pi]$ can be transformed in polynomial time into an equivalent QBF being in prenex form $\exists W_1 \forall W_2\, \phi$. Furthermore, hardness is a direct consequence of Proposition 6. □

This theorem shows that logic programs can be faithfully reduced to disjunctive logic programs in polynomial time. Note that the reduction presented in [19] does not meet this property.

The following result can be shown similarly to Theorem 5 (by invoking Corollary 1).

Theorem 6. *We have the following complexity results:*

1. *Deciding whether a formula ϕ is a brave consequence of a logic program Π is Σ_2^P-complete.*
2. *Deciding whether ϕ is a skeptical consequence of Π is Π_2^P-complete.*

Concerning full equilibrium logic, the following result was shown in [25]:

Proposition 7. *Deciding whether a formula has at least one equilibrium model is Σ_2^P-hard.*

Using our QBF-encoding for equilibrium logic (Theorem 1), we get a matching upper bound as follows:

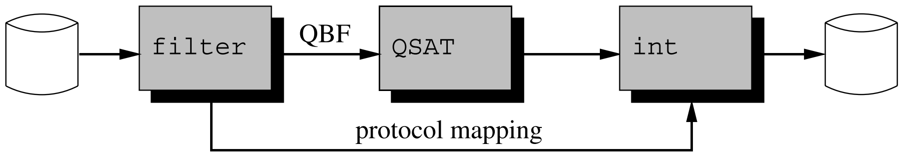


Fig. 1. Architecture to use different QBF-solvers.

Theorem 7. *Deciding whether a formula has at least one equilibrium model is Σ_2^P-complete.*

Finally, we deal with the complexity of deciding whether two logic programs are strongly equivalent.

The following lemma is a direct consequence of Proposition 3 and Lemma 2.

Lemma 4. *Programs Π_1 and Π_2 are strongly equivalent iff $(\mathcal{T}_{HT}[\hat{\Pi}_1] \leftrightarrow \mathcal{T}_{HT}[\hat{\Pi}_2])$ is a tautology of classical logic, where $\hat{\Pi}_i = \{B(r) \rightarrow H(r) \mid r \in \Pi_i\}$ $(i = 1, 2)$.*

Theorem 8. *Deciding whether two logic programs are strongly equivalent is in* coNP.

6 Implementation

Our methodology for considering encodings of different reasoning tasks into quantified Boolean formulas is motivated by the availability of several practically efficient QBF-solvers. Among the different tools, there is a propositional theorem-prover, `boole`, based on *binary decision diagrams* (the system can be downloaded from the Web at http://www.cs.cmu.edu/~modelcheck/bdd.html), a system using a generalized resolution principle [15], several provers implementing an extended Davis-Putnam procedure [2,11,16,9,27], as well as a distributed algorithm running on a PC-cluster [9]. With the exception of `boole`, these tools do not accept arbitrary QBFs, but require the input formula to be in prenex conjunctive normal form. To avoid an exponential increase of formula size, *structure preserving normal form translations* [4,26] can be used to translate a general QBF into the required normal form. In contrast to the usual normal form translation based on distributivity laws, structure preserving normal form translations introduce new labels for subformula occurrences and are polynomial in the length of the input formula.

The translations discussed in Sections 3 and 4 have been implemented as a special module of the reasoning system [7,6,5,3], which is a prototype tool for solving several nonmonotonic reasoning tasks based on reductions to QBFs.

The general architecture of is depicted in Figure 1. consists of three parts, namely the `filter` program, a QBF-evaluator, and the interpreter `int`. The input filter translates the given problem description (like, e.g., a logic program and a specified reasoning task) into the corresponding quantified Boolean formula, which is then sent to the QBF-evaluator. The current version of provides interfaces to most of the sequential QBF-solvers mentioned above. For the solvers requiring prenex normal form, the QBFs are translated into structure preserving normal form. The result of the QBF-evaluator is interpreted by `int`. Depending on the capabilities of the employed QBF-evaluator, `int` provides an explanation in terms of the underlying problem instance (e.g., listing all stable models of a given logic program). This task relies on a protocol mapping of internal variables of the generated QBF into concepts of the problem description which is provided by `filter`.

Finally, we note that, due to Theorems 4 and 8, testing the satisfiability of formulas in the logic of here-and-there, or testing the equivalence of two logic programs, can also be implemented using satisfiability checkers like SATZ [17], SATO [28] or RELSAT [1].

7 Concluding Remarks

We have shown how to encode reasoning problems in the nonmonotonic system of equilibrium logic by means of quantified Boolean formulas and thereby how to implement equilibrium logic using QBF solvers. Since this logic generalises the semantics of stable models for logic programs with nested expressions, our QBF reduction applies *a fortiori* here as well, yielding a polynomial translation. The earlier QBF reduction of ordinary (disjunctive) logic programs, given in [7], can be obtained as a special case.

Analysing these reduction methods yields complexity results for equilibrium logic, extending those obtained in [25]. New complexity results for nested expressions, extending those of [8] for disjunctive logic programs, have also been obtained. As a further corollary we have also derived complexity results for deciding whether two logic programs are strongly equivalent. Notice that, since the Lloyd-Topor-Semantics [20] for logic programs can be expressed in terms of logic programs with nested expressions, as shown by [19], our encoding is also applicable to this formalism.

In future work we hope to study further refinements of the basic encoding, in particular to show how programs with nested expressions can be efficiently reduced to equivalent disjunctive programs (in a suitably extended sense). Another topic is how to deal with a second (strong) negation operator, present in equilibrium logic and in the programs of [19]. Lastly, it is hoped to study examples of practical reasoning problems using an implementation of the formalisms studied here in the system.

A Proof of Lemma 2

The proof proceeds by induction on the logical complexity $lc(\phi)$ of ϕ. First, observe that, for $I_H, I_T \subseteq V$, the pair $\langle I_H, I_T \rangle$ is an HT-interpretation iff $I_H \cup I'_T$ is a model of $V \leq V'$.

INDUCTION BASE. Assume $lc(\phi) = 0$. Then, ϕ is either an atom, or one of $\top$ or $\bot$. If ϕ is one of $\top$ or $\bot$, then the statement holds trivially. So, suppose that $\phi = v$, for some atom v. Then, $V = \{v\}$ and $\tau[v] = v$. Hence,

$$\mathcal{T}_{HT}[\phi] = (v \rightarrow v') \wedge v.$$

Assume that $\mathcal{I} = \langle I_H, I_T \rangle$ is an HT-model of ϕ, i.e., we have that $I_H \subseteq I_T$ and $v \in I_H$. From this, we immediately get $v \in I_T$, and therefore $v' \in I'_T$. So, $I_H \cup I'_T = \{v, v'\}$, which is clearly a model of $\mathcal{T}_{HT}[\phi]$. Conversely, if $I_H \cup I'_T$ is a model of $\mathcal{T}_{HT}[\phi] = (v \rightarrow v') \wedge v$, for some $I_H \subseteq V$ and $I'_T \subseteq V'$, then $v \in I_H$. Given that $I_H \cup I'_T$ is a model of $(v \rightarrow v')$, it follows that $\mathcal{I} = \langle I_H, I_T \rangle$ is an HT-interpretation, so, in virtue of $v \in I_H$, $\mathcal{I}$ is an HT-model of ϕ.

INDUCTION STEP. Assume $lc(\phi) > 0$, and let the statement hold for all formulas ψ such that $lc(\psi) < lc(\phi)$. We have to consider several cases, depending on the structure of ϕ. Due to space restrictions, we only show two cases; the other ones follow by similar arguments.

Assume that $\phi = (\phi_1 \vee \phi_2)$. Then, $\mathcal{T}_{HT}[\phi]$ is given by

$$(V \leq V') \wedge \tau[\phi_1 \vee \phi_2], \tag{5}$$

where $\tau[\phi_1 \vee \phi_2] = \tau[\phi_1] \vee \tau[\phi_2]$. By taking V_i as the set of variables occurring in ϕ_i $(i = 1, 2)$, it follows that (5) is classically equivalent to

$$(V \leq V') \wedge \Big(\big((V_1 \leq V'_1) \wedge \tau[\phi_1] \big) \vee \big((V_2 \leq V'_2) \wedge \tau[\phi_2] \big) \Big),$$

which represents $(V \leq V') \wedge (\mathcal{T}_{HT}[\phi_1] \vee \mathcal{T}_{HT}[\phi_2])$.

Suppose now that $\mathcal{I} = \langle I_H, I_T \rangle$ is an HT-model of $\phi = (\phi_1 \vee \phi_2)$. We get that $\mathcal{I}$ is an HT-model of ϕ_1 or of ϕ_2. Without loss of generality, we assume that $\mathcal{I}$ is an HT-model of ϕ_1. Since $lc(\phi_1) < lc(\phi)$, by induction hypothesis it follows that $I_H \cup I'_T$ is a model of $\mathcal{T}_{HT}[\phi_1]$. Hence, $I_H \cup I'_T$ is also a model of $\mathcal{T}_{HT}[\phi_1] \vee \mathcal{T}_{HT}[\phi_2]$. Furthermore, since $\mathcal{I}$ is an HT-interpretation, $I_H \cup I'_T$ is a model of $V \leq V'$. It follows that $I_H \cup I'_T$ is a model of $(V \leq V') \wedge (\mathcal{T}_{HT}[\phi_1] \vee \mathcal{T}_{HT}[\phi_2])$. Since the last formula is equivalent to $\mathcal{T}_{HT}[\phi]$, we obtain that $I_H \cup I'_T$ is a model of $\mathcal{T}_{HT}[\phi]$. The converse direction follows in essentially the same way.

Now assume that $\phi = \neg\psi$. By definition of $\tau[\cdot]$,

$$\mathcal{T}_{HT}[\phi] = (V \leq V') \wedge \neg\tau[\psi] \wedge \neg\psi'.$$

Suppose that $\mathcal{I} = \langle I_H, I_T \rangle$ is an HT-model of $\phi = \neg\psi$. Then, $\nu_{\mathcal{I}}(u, \psi) = 0$, for each $u \in \{H, T\}$. Since $lc(\psi) < lc(\phi)$, and given that $\nu_{\mathcal{I}}(H, \psi) = 0$, the induction hypothesis implies that $I_H \cup I'_T$ is not a model of $\mathcal{T}_{HT}[\psi] = (V \leq V') \wedge \tau[\psi]$. But $I_H \cup I'_T$ is a model of $(V \leq V')$ (since $\mathcal{I}$ is an HT-interpretation), so $I_H \cup I'_T$ is not a model of $\tau[\psi]$. It follows that $I_H \cup I'_T$ is a model of $\neg\tau[\psi]$. On the other hand, given that $\nu_{\mathcal{I}}(T, \psi) = 0$, Part 1 of Proposition 1 implies that $\nu_{I_T}(\psi) = 0$. Hence, a simple renaming yields that $I_H \cup I'_T$ is a model of $\neg\psi'$. Finally, $I_H \cup I'_T$ is a model of $V \leq V'$ because $\mathcal{I}$ is an HT-interpretation. We conclude that $I_H \cup I'_T$ is a model of $\mathcal{T}_{HT}[\phi]$. The proof of the converse direction proceeds analogously.

References

1. R. Bayardo and R. Schrag. Using CSP Look-Back Techniques to Solve Real-World SAT Instances. In *Proc. AAAI-97*, pp. 203–208, 1997.

2. M. Cadoli, A. Giovanardi, and M. Schaerf. An Algorithm to Evaluate Quantified Boolean Formulae. In *Proc. AAAI-98*, pp. 262–267, 1998.
3. J. Delgrande, T. Schaub, H. Tompits, and S. Woltran. On Computing Solutions to Belief Change Scenarios. In *Proc. ECSQARU-01*, pp. 510–521, 2001.
4. E. Eder. *Relative Complexities of First-Order Calculi*. Vieweg Verlag, 1992.
5. U. Egly, T. Eiter, R. Feldmann, V. Klotz, S. Schamberger, H. Tompits, and S. Woltran. On Mechanizing Modal Nonmonotonic Logics. In *Proc. DGNMR-01*, pp. 44–53, 2001.
6. U. Egly, T. Eiter, V. Klotz, H. Tompits, and S. Woltran. Computing Stable Models with Quantified Boolean Formulas: Some Experimental Results. In *Proc. AAAI Spring Symposium-01*, pp. 53–59, 2001.
7. U. Egly, T. Eiter, H. Tompits, and S. Woltran. Solving Advanced Reasoning Tasks Using Quantified Boolean Formulas. In *Proc. AAAI-00*, pp. 417–422, 2000.
8. T. Eiter and G. Gottlob. On the Computational Cost of Disjunctive Logic Programming: Propositional Case. *Ann. of Math. and Artificial Intelligence*, 15(3–4):289–323, 1995.
9. R. Feldmann, B. Monien, and S. Schamberger. A Distributed Algorithm to Evaluate Quantified Boolean Formulas. In *Proc. AAAI-00*, pp. 285–290, 2000.
10. M. Gelfond and V. Lifschitz. Classical Negation in Logic Programs and Disjunctive Databases. *New Generation Computing*, 9:365–385, 1991.
11. E. Giunchiglia, M. Narizzano, and A. Tacchella. QUBE: A System for Deciding Quantified Boolean Formulas Satisfiability. In *Proc. IJCAR-01*, pp. 364–369, 2001.
12. K. Gödel. Zum intuitionistischen Aussagenkalkül. *Anzeiger der Akademie der Wissenschaften in Wien*, pp. 65–66, 1932.
13. A. Heyting. Die formalen Regeln der intuitionistischen Logik. *Sitz. Berlin*, pp. 42–56, 1930.
14. H. Kautz and B. Selman. Planning as Satisfiability. In *Proc. ECAI-92*, pp. 359–363, 1992.
15. H. Kleine-Büning, M. Karpinski, and A. Flögel. Resolution for Quantified Boolean Formulas. *Information and Computation*, 117(1):12–18, 1995.
16. R. Letz. Advances in Decision Procedures for Quantified Boolean Formulas. In *Proc. IJCAR-01 Workshop on Theory and Applications of Quantified Boolean Formulas*, pp. 55–64, 2001.
17. C. M. Li and Anbulagan. Heuristics Based on Unit Propagation for Satisfiability Problems. In *Proc. IJCAI-97*, pp. 366–371, 1997.
18. V. Lifschitz, D. Pearce, and A. Valverde. Strongly Equivalent Logic Programs. *ACM Transactions on Computational Logic*, 2(4), 2001. To appear.
19. V. Lifschitz, L. Tang, and H. Turner. Nested Expressions in Logic Programs. *Ann. of Math. and Artificial Intelligence*, 25(3-4):369–389, 1999.
20. J. Lloyd and R. Topor. Making Prolog More Expressive. *J. of Logic Progr.*, 3:225–240, 1984.
21. J. Lukasiewicz. Die Logik und das Grundlagenproblem. *Les Entretiens de Zürich sue les Fondements et la Méthode des Sciences Mathématiques*, 6-9, 12 (1938), 1941.
22. D. Pearce. A New Logical Characterisation of Stable Models and Answer Sets. In *Non-Monotonic Extensions of Logic Programming*, pp. 57–70. Springer, 1997.
23. D. Pearce. From Here to There: Stable Negation in Logic Programming. In *What is Negation?* Kluwer, 1999.
24. D. Pearce, I. de Guzmán, and A. Valverde. A Tableau Calculus for Equilibrium Entailment. In *Proc. TABLEAUX 2000*, pp. 352–367, 2000.
25. D. Pearce, I. de Guzmán, and A. Valverde. Computing Equilibrium Models Using Signed Formulas. In *Proc. Computational Logic 2000*, pp. 688–702, 2000.
26. D. A. Plaisted and S. Greenbaum. A Structure Preserving Clause Form Translation. *J. of Symbolic Computation*, 2(3):293–304, 1986.
27. J. Rintanen. Improvements to the Evaluation of Quantified Boolean Formulae. In *Proc. IJCAI-99*, pp. 1192–1197, 1999.
28. H. Zhang. SATO: An Efficient Propositional Prover. In *Proc. CADE-97*, pp. 272–275, 1997.

A Context-Free Grammar Representation for Normal Inhabitants of Types in $\mathbf{TA}_\lambda$

Sabine Broda and Luís Damas

DCC-FC & LIACC, Universidade do Porto,
R. do Campo Alegre, 823, 4150-180 Porto, Portugal

Abstract. In [10] it was shown that it is possible to describe the set of normal inhabitants of a given type τ, in the standard simple type system, using an infinitary extension of the concept of context-free grammar, which allows for an infinite number of non-terminal symbols as well as production rules. The set of normal inhabitants of τ corresponds then to the set of terms generated by this, possibly infinitary, grammar plus all terms obtained from those by η-reduction. In this paper we show that the set of normal inhabitants of a type τ can in fact be described using a standard (finite) context-free grammar, and more interestingly that normal inhabitants of types with the same structure are described by identical context-free grammars, up to renaming of symbols.

1 Introduction

The relation of terms and types in the system $\mathbf{TA}_\lambda$ of simply typed λ-calculus has been a major subject of study over the last decades. The interest in this area is due to its importance to areas of mathematical logic and more recently to computer science and artificial intelligence. In fact, systems of lambda calculus are of importance for most knowledge representation theories and in particular for several systems for natural language processing. On the other hand, it is well known that there exists a direct correspondence (cf. [8]), via Curry-Howard isomorphism, between $\mathbf{TA}_\lambda$ and the implicational fragment of intuitionistic propositional logic, $\mathcal{P}(\rightarrow)$. As such, a type τ can be assigned to some λ-term M[1] if and only if τ is a provable formula/theorem of $\mathcal{P}(\rightarrow)$. Furthermore, every (normal) term to which τ can be assigned, i.e. every (normal) inhabitant of τ, represents a (normal) proof of τ in the natural deduction system for the implicational fragment of intuitionistic logic. Thus, the study of inhabitation in $\mathbf{TA}_\lambda$ corresponds directly to the study of provability in $\mathcal{P}(\rightarrow)$. The decision problem for inhabitation as well as the infiniteness problem of the set of normal inhabitants have been shown to be polynomial-space complete, respectively by Statman in [9] and by Hirokawa in [7]. In [2], Ben-Yelles defined an algorithm, also described in [6], to count and list the set `Nhabs`(τ) of normal inhabitants of a type τ. An adaptation of his algorithm for the subsystem λI and similar algorithms for principal normal inhabitants in both system were presented in [3]. Other algorithms, which

[1] i.e. M is an inhabitant of τ

for every type produce a normal inhabitant or guarantee the non-existence of inhabitants, were given in [5].

In [10] it was shown that it is possible to describe the set of normal inhabitants of a type, using an infinitary extension of the concept of context-free grammar, which allows for an infinite number of non-terminal symbols as well as production rules. The set of normal inhabitants of τ corresponds then to the set of terms generated by this, possibly infinitary, grammar plus all terms obtained from those by η-reduction. In this paper we show that the set of normal inhabitants of a type τ can in fact be described using a standard (finite) context-free grammar, and more interestingly that types with the same structure share the same grammar, up to renaming of symbols. This result partly explains an observation made by Hindley in [6], where he pointed out that "when we ask for the number of normal inhabitants of a type, the answer is often finite and interesting patterns show up which are still not completely understood." On the other hand, it provides us with a new angle to take into account in the study of the relation between the structure of formulas and their proofs. The definition of a common grammar-scheme for a given type structure, is based on an alternative representation for types, introduced in [4], which gives us a better insight on the nature of a type's structure and its relation to the structure of the set of its normal inhabitants.

2 Background

We assume familiarity with the basic notions in λ-calculus and use standard notation from [1] and [6]. Our notation differs from that in [1], since we denote type-variables (atoms) by "A,B,C,..."and arbitrary types by lower-case Greek letters. This choice results from the standard notation for context-free grammars and from the fact that we will use an annotated version of the atoms in a type as non-terminal symbols of the context-free grammar which describes the normal inhabitants of this type. For type assignment we consider the system $\mathbf{TA}_\lambda$ of simply typed λ-calculus à la Curry (for an introduction see [6] or [1]).

In this section we recall the definition of an alternative tree-like representation for types, called formula trees, first introduced in [4], which gives us an exact idea on the different stages during the construction of normal proofs/inhabitants of a formula/type. In fact, the formula tree of a type τ defines some kind of hierarchy over the primitive parts of τ which can be used to construct proofs of τ represented by proof trees.

2.1 Formula Trees and Proof Trees

Formula trees are trees whose nodes consist of primitive parts which are of either one of the following forms (P1), (P2) or (P3):

$$\text{(P1):}\quad \begin{array}{c} | \\ A \end{array} \qquad \text{(P2):}\quad \begin{array}{ccc} & A & \\ / & & \backslash \\ A_1 & \cdots & A_n \end{array} \quad (n \geq 1) \qquad \text{(P3):}\quad \begin{array}{c} A \\ | \end{array}$$

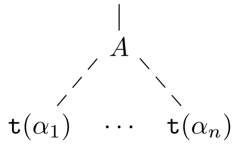
Definition 1. *A tree with primitive parts as nodes is a* formula tree *iff*

– *the root of the tree is of form (P1);*
– *every internal node of the tree is of form (P2);*
– *every leaf of the tree is of form (P2) or (P3).*

The following algorithm computes the formula tree $\texttt{tree}(\varphi)$ of a type φ. We use dashed lines for the edges of the formula tree in order to distinguish them from the edges in the primitive parts (nodes) of the tree. Note that every type φ can be written uniquely in the form $\varphi = \alpha_1 \to \ldots \to \alpha_n \to A$, where A is an atom and $n \geq 0$. The algorithm is given by the following.

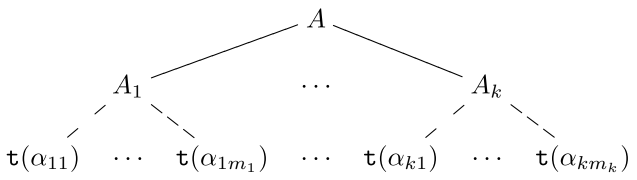
– If $n = 0$, i.e. $\varphi \equiv A$, then $\texttt{tree}(\varphi) = \begin{array}{c} | \\ A \end{array}$.

– If $n \geq 1$, then $\texttt{tree}(\varphi) =$



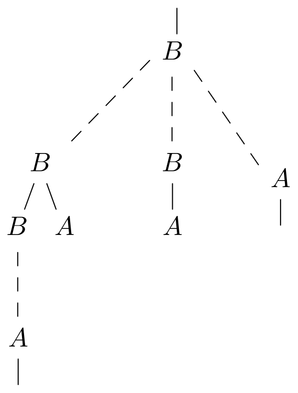
where $\texttt{t}(A) = \begin{array}{c} A \\ | \end{array}$

and for $k \geq 1$ and $m_1, \ldots, m_k \geq 0$ we define $\texttt{t}((\alpha_{11} \to \ldots \to \alpha_{1m_1} \to A_1) \to \ldots \to (\alpha_{k1} \to \ldots \to \alpha_{km_k} \to A_k) \to A) =$

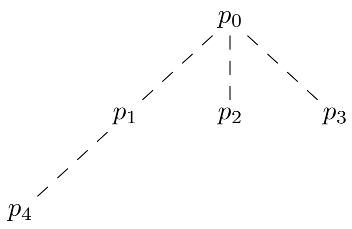


Note that one can easily define an inverse algorithm, which given a formula tree $\texttt{FT}$ computes the unique type φ such that $\texttt{tree}(\varphi) = \texttt{FT}$.

Example 1. The formula $((A \to B) \to A \to B) \to (A \to B) \to A \to B$ has the following formula tree



i.e.



with primitive parts

$$p_0 = \begin{array}{c} | \\ B \end{array} \qquad p_1 = \begin{array}{c} B \\ / \ \backslash \\ B \quad A \end{array} \qquad p_2 = \begin{array}{c} B \\ | \\ A \end{array} \qquad p_3 = \begin{array}{c} A \\ | \end{array} \qquad p_4 = \begin{array}{c} A \\ | \end{array} .$$

Definition 2. *A proof tree built from a formula tree* `tree`(φ) *is obtained by joining primitive parts of* `tree`(φ) *identifying/overlapping (different) occurrences of the same type variable. We call it a* valid proof tree *if*

– *all leafs in the proof tree are of form (P3);*
– *whenever a primitive part p_i occurs beneath some primitive part p_j in the formula tree* `tree`(φ)*, then above every occurrence of p_i in the proof tree there is at least one occurrence of p_j.*

Note that the formula tree of φ defines some kind of hierarchy over its primitive parts, which has to be respected by every valid proof tree built from `tree`(φ). In particular, this means that the root of every valid proof tree built from `tree`(φ) is the root of `tree`(φ).

Example 2. A valid proof tree for the formula in example 1 is

$$\begin{array}{c} p_0 \\ p_2 \\ p_3 \end{array} \quad \text{which looks like} \quad \begin{array}{c} | \\ B \\ | \\ A \\ | \end{array}$$

On the other hand,

$$\begin{array}{c} p_0 \\ p_2 \\ p_4 \end{array}$$

is no valid proof tree (though it would have the same appearance), since it does not respect the hierarchy given by the formula tree

$$\begin{tikzcd} & & p_0 \arrow[dl, dash] \arrow[d, dash] \arrow[dr, dash] & \\ & p_1 \arrow[dl, dash] & p_2 & p_3 \\ p_4 & & & \end{tikzcd}$$

of $((A \to B) \to A \to B) \to (A \to B) \to A \to B$, which requires that p_4 should only be used beneath occurrences of both p_0 and p_1. Other valid proof trees are

$$\begin{array}{c} p_0 \\ p_1 \\ p_2 \ p_3 \\ p_4 \end{array} \quad \text{and} \quad \begin{array}{c} p_0 \\ p_1 \\ p_2 \ p_3 \\ p_3 \end{array} \quad \text{both with appearance} \quad \begin{array}{c} | \\ B \\ / \ \backslash \\ B \quad A \\ | \quad\ | \\ A \\ | \end{array} .$$

2.2 Proof Trees and (Long) Normal Inhabitants

A β-normal inhabitant M of a type φ is called a long normal inhabitant of φ iff every variable-occurrence z in M is followed by the longest sequence of arguments allowed by its type, i.e. iff each component with form $(zP_1 \dots P_n)$, $(n \geq 0)$ that is not in a function position has atomic type. The (finite) set of all terms obtained by η-reducing a λ-term M is called the η-family of M and denoted by $\{M\}_\eta$. It has been shown (cf. [2], [6]) that the η-families of the long normal inhabitants of φ partition the set of normal inhabitants of φ into non-overlapping finite subsets, each η-family containing just one long member. Furthermore, Ben-Yelles (cf. [2], [6]) showed that every normal inhabitant of a type φ can be η-expanded to one unique (up to α-conversion) long normal inhabitant of φ. A simple expansion-algorithm can be found in [6]. Thus, when seeking for normal inhabitants of a type it is sufficient to compute the set of its long normal inhabitants from which all normal inhabitants can be obtained by η-reduction.

In [4] an algorithm was defined which given a long inhabitant M of a type φ computes a valid proof tree $\mathtt{PT}(M)$ of $\mathtt{tree}(\varphi)$.

Proposition 1 (in [4]). *If M is a long normal inhabitant of a type φ, then $\mathtt{PT}(M)$ is a valid proof tree built from $\mathtt{tree}(\varphi)$.*

Also, an *inverse* algorithm was given in [4], which given the formula tree of a type φ, $\mathtt{t} = \mathtt{tree}(\varphi)$, and any valid proof tree PT built from it, computes a set $\mathtt{Terms}(\mathtt{PT})$ of long closed normal inhabitants of φ.

Proposition 2 (in [4]). *Let PT be a valid proof tree built from the formula tree of some type φ. Then every member of $\mathtt{Terms}(\mathtt{PT})$ is a closed long normal inhabitant of φ. Furthermore, the two algorithms are complementary in the sense that for every closed long normal inhabitant M of φ there is $M \in \mathtt{Terms}(\mathtt{PT}(M))$.*

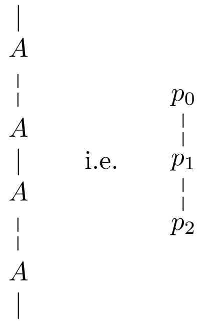
To sum up, every normal inhabitant of a type φ corresponds to one valid proof tree built from $\mathtt{tree}(\varphi)$, every valid proof tree built from $\mathtt{tree}(\varphi)$ corresponds to a finite set of normal inhabitants of φ, and distinct valid proof trees correspond to distinct and disjoint sets of normal inhabitants.

3 A Context-Free Grammar for $\mathtt{Nhabs}(\tau)$

3.1 Using Term-Schemes Instead of Proof Trees

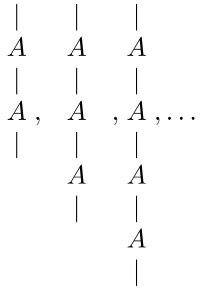
Up to now we saw that the set of normal inhabitants of a type τ can be computed by generating all proof trees of τ, applying the algorithm Terms and collecting all terms which can be obtained from these by η-reduction. In the following we will introduce the notion of term-scheme for λ-terms, in such a way that for each proof tree PT there is a term-scheme T such that $\mathtt{Terms}(\mathtt{PT}) = \mathcal{C}(T)$, where $\mathcal{C}(T)$ is the set of λ-terms represented by T. In a term-scheme T different variables of the same type are represented with the same name, and the corresponding set of λ-terms $\mathcal{C}(T)$ can be obtained from T by instantiating them to different names, while respecting scoping rules.

Example 3. The formula $((A \to A) \to A) \to A$ with formula tree



with primitive parts $p_0 = \begin{array}{c} | \\ A \end{array}$, $p_1 = \begin{array}{c} A \\ | \\ A \end{array}$ and $p_2 = \begin{array}{c} A \\ | \end{array}$ has the following valid proof trees $\mathtt{PT}_1$, $\mathtt{PT}_2$, $\mathtt{PT}_3$, …

$$\begin{array}{c} p_0 \\ p_1 \\ p_2 \end{array}, \quad \begin{array}{c} p_0 \\ p_1 \\ p_1 \\ p_2 \end{array}, \quad \begin{array}{c} p_0 \\ p_1 \\ p_1 \\ p_1 \\ p_2 \end{array}, \ldots \qquad \text{i.e.}$$



For these proof trees the algorithm `Terms` produces respectively the following sets of long normal inhabitants:

$$\begin{aligned}
\mathtt{Terms}(\mathtt{PT}_1) &= \{\ \lambda x_1.x_1(\lambda x_2.x_2)\ \}, \\
\mathtt{Terms}(\mathtt{PT}_2) &= \{\ \lambda x_1.x_1(\lambda x_2.x_1(\lambda x_2'.x_2)),\ \lambda x_1.x_1(\lambda x_2.x_1(\lambda x_2'.x_2'))\ \}, \\
\mathtt{Terms}(\mathtt{PT}_2) &= \{\ \lambda x_1.x_1(\lambda x_2.x_1(\lambda x_2'.x_1(\lambda x_2''.x_2))), \\
&\qquad \lambda x_1.x_1(\lambda x_2.x_1(\lambda x_2'.x_1(\lambda x_2''.x_2'))), \\
&\qquad \lambda x_1.x_1(\lambda x_2.x_1(\lambda x_2'.x_1(\lambda x_2''.x_2'')))\ \}\ , \ldots
\end{aligned}$$

The previous example illustrates the fact that, although a proof tree may in fact represent more than one, but a finite number, of long normal inhabitants of a type, all those inhabitants follow a common pattern which will be captured by the notion of term-scheme. As such, for every valid proof tree `PT` for a type τ, there will be a term-scheme representing exactly the terms computed by `Terms(PT)`.

Definition 3. *Syntactically term-schemes are λ-terms. The (finite) set of λ-terms represented by a term-scheme T with variables $x_1, \ldots, x_n$ is defined by $\mathcal{C}(T) = T^{0\ldots0}_{x_1\ldots x_n}$ as follows.*

– $x_i{}^{k_1\ldots0\ldots k_n}_{x_1\ldots x_i\ldots x_n} = \{\overline{x_i}\}$;[2] $\quad -x_i{}^{k_1\ldots k_i\ldots k_n}_{x_1\ldots x_i\ldots x_n} = \{x_i, x_i', x_i'', \ldots, x_i^{k_i-1}\}, k_i \geq 1$;

[2] This case corresponds to the renaming of free occurrences of variables in T.

- $(ST)^{k_1\dots k_n}_{x_1\dots x_n} = \{S_iT_j \mid S_i \in S^{k_1\dots k_n}_{x_1\dots x_n}, T_j \in T^{k_1\dots k_n}_{x_1\dots x_n}\}$;
- $(\lambda x_i.T)^{k_1\dots k_n}_{x_1\dots x_n} = \{\lambda x_i^{k_i}.T_i \mid T_i \in T^{k_1\dots(k_i+1)\dots k_n}_{x_1\dots x_i\dots x_n}\}$.

Example 4. The scheme $T = \lambda xy.x(\lambda xy.xy)(\lambda y.xy)$ represents the following set of λ-terms.

$$\begin{aligned}\mathcal{C}(T) = \{\ & \lambda xy.x(\lambda x'y'.xy)(\lambda y'.xy),\\ & \lambda xy.x(\lambda x'y'.xy)(\lambda y'.xy'),\\ & \lambda xy.x(\lambda x'y'.xy')(\lambda y'.xy),\\ & \lambda xy.x(\lambda x'y'.xy')(\lambda y'.xy'),\\ & \lambda xy.x(\lambda x'y'.x'y)(\lambda y'.xy),\\ & \lambda xy.x(\lambda x'y'.x'y)(\lambda y'.xy'),\\ & \lambda xy.x(\lambda x'y'.x'y')(\lambda y'.xy),\\ & \lambda xy.x(\lambda x'y'.x'y')(\lambda y'.xy')\ \}\end{aligned}$$

Example 5. The sets of long normal inhabitants corresponding to the proof trees in example 3 can respectively be represented by the following term-schemes:

$\texttt{Terms}(\texttt{PT}_1) = \mathcal{C}(\lambda x_1.x_1(\lambda x_2.x_2))$, $\texttt{Terms}(\texttt{PT}_2) = \mathcal{C}(\lambda x_1.x_1(\lambda x_2.x_2))$, $\texttt{Terms}(\texttt{PT}_3) = \mathcal{C}(\lambda x_1.x_1(\lambda x_2.x_1(\lambda x_2.x_2)))$, ...

3.2 The Context-Free Grammar $\texttt{G}_\tau$

In this subsection we will, for every type τ, define a context-free grammar $\texttt{G}_\tau$ which generates a set of term-schemes, that correspond exactly to the long normal inhabitants of τ.

Definition 4. *Let τ be a type, with atoms $A_1, \dots, A_m$ and such that $\texttt{tree}(\tau)$ has primitive parts $p_0, \dots, p_n$, where p_0 represents the root of $\texttt{tree}(\tau)$. Let $\texttt{G}(\tau) = (T, N, R, S)$ be the context-free grammar with*

- *set of terminal symbols $T = \{(,), \lambda, ., x_1, \dots, x_n\}$,*
- *set of non-terminal symbols $N = \{S\} \cup \{A_i^P \mid 1 \le i \le m, P \in 2^{\{1,\dots,n\}}\}$*
- *start symbol S*

and such that R is the smallest set satisfying the following conditions.

- *if $p_0 = \overset{|}{A_s}$ has descendents $p_{i_1}, \dots, p_{i_t}$ in $\texttt{tree}(\tau)$, then R has exactly one production rule for S which is*

$$S \to \lambda x_{i_1} \dots x_{i_t}.A_s^{\{i_1,\dots,i_t\}}$$

- *whenever a non-terminal symbol A_j^P appears on the right side of a production rule in R, then for every $i \in P$ such that p_i is of the form $\underset{|}{A_j}$ there is a rule in R of the form $A_j^P \to x_i$*

– *whenever a non-terminal symbol A_j^P appears on the right side of a production rule in R, then for every $i \in P$ such that p_i is of the form*

$$\begin{tikzcd}[column sep=small] & A_j \arrow[dl, dash] \arrow[dr, dash] & \\ A_{j_1} & \cdots & A_{j_s} \end{tikzcd}$$

and such that in $\texttt{tree}(\tau)$ *each A_{j_l} has descendents $p_{j_l^1}, \ldots, p_{j_l^{t_l}}$, there is a rule in R of the form*

$$A_j^P \to x_i(\lambda x_{j_1^1} \ldots x_{j_1^{t_1}}.A_{j_1}^{P_1}) \ldots (\lambda x_{j_s^1} \ldots x_{j_s^{t_s}}.A_{j_s}^{P_s})$$

where $P_l = P \cup \{j_l^1, \ldots, j_l^{t_l}\}$, for $1 \leq l \leq s$.

Example 6. For α as in example 1, there is $\mathtt{G}_\alpha = (T, N, R, S)$ with

– $T = \{(,), \lambda, ., x_1, x_2, x_3, x_4\}$,
– $N = \{S\} \cup \{A^P \mid P \subseteq \{1,2,3,4\}\} \cup \{B^P \mid P \subseteq \{1,2,3,4\}\}$,
– and production rules

$$\begin{array}{ll}
S & \to \lambda x_1 x_2 x_3.\ B^{123} \\
B^{123} & \to x_1(\lambda x_4.\ B^{1234})(\ A^{123}) \\
B^{123} & \to x_2(\ A^{123}) \\
B^{1234} & \to x_1(\lambda x_4.\ B^{1234})(\ A^{1234}) \\
B^{1234} & \to x_2(\ A^{1234}) \\
A^{123} & \to x_3 \\
A^{1234} & \to x_3 \\
A^{1234} & \to x_4
\end{array}$$

This grammar generates the following infinite set of term-schemes.

$$\begin{array}{rl}
\mathcal{L}(\mathtt{G}_\alpha) = \{ & \lambda x_1 x_2 x_3.x_2 x_3, \\
& \lambda x_1 x_2 x_3.x_1(\lambda x_4.x_2 x_3)x_3, \\
& \lambda x_1 x_2 x_3.x_1(\lambda x_4.x_2 x_4)x_3, \\
& \lambda x_1 x_2 x_3.x_1(\lambda x_4.x_1(\lambda x_4.x_2 x_3)x_3)x_3, \\
& \lambda x_1 x_2 x_3.x_1(\lambda x_4.x_1(\lambda x_4.x_2 x_3)x_4)x_3, \\
& \lambda x_1 x_2 x_3.x_1(\lambda x_4.x_1(\lambda x_4.x_2 x_4)x_3)x_3, \\
& \lambda x_1 x_2 x_3.x_1(\lambda x_4.x_1(\lambda x_4.x_2 x_4)x_4)x_3, \ldots \}
\end{array}$$

Thus,

$$\begin{aligned}
\texttt{Nhabs}(\alpha) = \{\, & M \mid \exists T \in \mathcal{L}\ \exists N \in \mathcal{C}(T) \text{ such that } N \to_\eta M\} \\
= \{\, & \lambda x_1 x_2 x_3 . x_2 x_3, \\
& \lambda x_1 x_2 . x_2, \\
& \lambda x_1 x_2 x_3 . x_1 (\lambda x_4 . x_2 x_3) x_3, \\
& \lambda x_1 x_2 x_3 . x_1 (\lambda x_4 . x_2 x_4) x_3, \\
& \lambda x_1 x_2 . x_1 (\lambda x_4 . x_2 x_4), \\
& \lambda x_1 x_2 x_3 . x_1 x_2 x_3, \\
& \lambda x_1 x_2 . x_1 x_2, \\
& \lambda x_1 x_2 x_3 . x_1 (\lambda x_4 . x_1 (\lambda x_4' . x_2 x_3) x_3) x_3, \\
& \lambda x_1 x_2 x_3 . x_1 (\lambda x_4 . x_1 (\lambda x_4' . x_2 x_3) x_4) x_3, \\
& \lambda x_1 x_2 x_3 . x_1 (x_1 (\lambda x_4' . x_2 x_3)) x_3, \\
& \lambda x_1 x_2 x_3 . x_1 (\lambda x_4 . x_1 (\lambda x_4' . x_2 x_4) x_3) x_3, \\
& \lambda x_1 x_2 x_3 . x_1 (\lambda x_4 . x_1 (\lambda x_4' . x_2 x_4') x_3) x_3, \\
& \lambda x_1 x_2 x_3 . x_1 (\lambda x_4 . x_1 x_2 x_3) x_3, \dots \}
\end{aligned}$$

3.3 Correctness of $\texttt{G}_\tau$

In the following we describe an algorithm which, given a type τ and a valid proof tree PT built from $\texttt{tree}(\tau)$, computes a term-scheme $T = \texttt{TS}(\texttt{PT})$ representing the set of long normal inhabitants that corresponds to PT.

Consider the root $\begin{array}{c}|\\A\end{array}$ of PT and let $p_1, \dots, p_n$, with $n \geq 0$, be the descendents of the corresponding occurrence of A in $\texttt{tree}(\tau)$.

- If PT is of the form $\begin{array}{c}|\\A\\|\end{array}$, then take the corresponding primitive part $p_i = \begin{array}{c}A\\|\end{array}$ in $\texttt{tree}(\tau)$ and let $\texttt{TS}(\texttt{PT}) = \lambda x_1 \dots \lambda x_n . x_i$.
- Otherwise, PT is of the form

$$\begin{tikzcd}[column sep=small]
 & {} \arrow[d, dash] & \\
 & A \arrow[dl, dash] \arrow[dr, dash] & \\
\texttt{PT}_1 & \cdots & \texttt{PT}_m
\end{tikzcd}$$

with $m \geq 1$, and such that for $j = 1, \dots, m$ the root of $\texttt{PT}_j$ is $\begin{array}{c}|\\B_j\end{array}$. Consider the corresponding primitive part p_i of the form

$$\begin{tikzcd}[column sep=small]
 & A \arrow[dl, dash] \arrow[dr, dash] & \\
B_1 & \cdots & B_m
\end{tikzcd}$$

and let

$$\texttt{TS}(\texttt{PT}) = \lambda x_1 \dots \lambda x_n . x_i M_1 \dots M_m,$$

where for $1 \leq j \leq m$,

$$M_j \in \mathtt{TS}(\mathtt{PT}_j).$$

The following result establishes the correctness of the algorithm TS.

Proposition 3. *Let τ be a type and* PT *a valid proof tree built from* $\mathtt{tree}(\tau)$*. Then, modulo α-conversions,*

$$\mathcal{C}(\mathtt{TS}(\mathtt{PT})) = \mathtt{Terms}(\mathtt{PT}).$$

Proof The result should become quite clear after recalling the definition of the algorithm Terms in [4]. Given the formula tree of a type φ, $\mathtt{t} = \mathtt{tree}(\varphi)$, and any valid proof tree PT built from it, $\mathtt{Terms}(\mathtt{PT}) = \mathtt{Terms}(\mathtt{t}, 0, \mathtt{PT}, \emptyset)$ is defined as follows. In the following let L represent a set assigning variable names to primitive parts of t. Then, $\mathtt{Terms}(\mathtt{t}, k, \mathtt{PT}, L)$ is defined by the following.

Let $\overset{|}{A}$ be the root of PT and consider the corresponding occurrence of A in t with (direct) descendents $p_1, \ldots, p_n$, $n \geq 0$. Now, let $L' = L \cup \{(p_i, x_i^k) \,|\, 1 \leq i \leq n\}$.

- If PT is of the form $\overset{|}{\underset{|}{A}}$, then take the corresponding primitive part $p = \underset{|}{A}$ and let $\mathtt{Terms}(\mathtt{t}, k, \mathtt{PT}, L) = \{\lambda x_1^k \ldots \lambda x_n^k . x \,|\, (p, x) \in L'\}$.
- Otherwise, PT is of the form

  $$\begin{tikzcd} & \overset{|}{A} \arrow[dl, dash] \arrow[dr, dash] & \\ \mathtt{PT}_1 & \cdots & \mathtt{PT}_m \end{tikzcd}$$

  with $m \geq 1$, and such that for $i = 1, \ldots, m$ the root of $\mathtt{PT}_i$ is $\overset{|}{B_i}$. Consider the corresponding primitive part p of the form

  $$\begin{tikzcd} & A \arrow[dl, dash] \arrow[dr, dash] & \\ B_1 & \cdots & B_m \end{tikzcd}$$

  and let $\mathtt{Terms}(\mathtt{t}, k, \mathtt{PT}, L) = \{\lambda x_1^k \ldots \lambda x_n^k . x M_1 \ldots M_m \,|\, (p, x) \in L'$, $M_j \in \mathtt{Terms}(\mathtt{t}, k+1, \mathtt{PT}_j, L'), 1 \leq j \leq m\}$

Now note that, when erasing all superscripts of variables in the members of Terms(PT), one obtains exactly one term, which is TS(PT). On the other hand, the computation done by $\mathcal{C}$ is performed during the construction of Terms(PT): in fact, if an abstraction over a variable x occurs in the range of another abstraction over x, then the two occurrences of x receive different superscripts, the available superscripts for x are stored in the set L and whenever x is used, all possibilities (i.e. all possible superscripts) are explored. •

Proposition 4. *For any type τ the grammar $\mathtt{G}_\tau$ describes $\mathtt{Nhabs}(\tau)$ in the sense that*

$$\mathtt{Nhabs}(\tau) = \{M \mid \exists T \in \mathcal{L}(\mathtt{G}_\tau)\ \exists N \in \mathcal{C}(T) \textit{ such that } N \to_\eta M\}.$$

Proof Note, that the definition of $\mathtt{G}_\tau$ is essentially a description of the possibilities of constructing valid proof trees from $\mathtt{tree}(\tau)$. As such, the production rules for a non-terminal symbol $A^{i_1,\ldots,i_n}$ describe the possible ways of overlapping an occurrence of A with another occurrence of A in a primitive part p_i of $\mathtt{tree}(\tau)$. For this, p_i has to be available, corresponding to the condition $i \in \{i_1, ..., i_n\}$ and A has to be the root or p_i. Then, instead of generating valid proof trees, these are simultaneously translated to the corresponding term scheme they represent. •

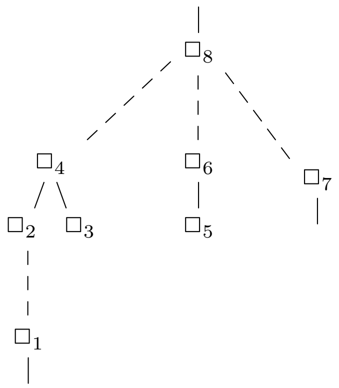
4 A Common Grammar Scheme for Types with the Same Structure

Given a type τ we represent the corresponding type-structure replacing occurrences of atoms by blank successively indexed boxes as illustrated in the following example.

Example 7. The type-structure corresponding to α from example 1 is

$$\Theta = ((\square_1 \to \square_2) \to \square_3 \to \square_4) \to (\square_5 \to \square_6) \to \square_7 \to \square_8$$

with formula tree



Possible instances, i.e types with this structure, are for example,

1. for $\square_1 = \square_2 = \square_3 = \square_5 = \square_7 = A$ and $\square_4 = \square_6 = \square_8 = B$ the type
$$\alpha_1 = ((A \to A) \to A \to B) \to (A \to B) \to A \to B$$
2. for $\square_1 = \square_2 = A$, $\square_3 = \square_7 = B$ and $\square_4 = \square_5 = \square_6 = \square_8 = C$ the type
$$\alpha_2 = ((A \to A) \to B \to C) \to (C \to C) \to B \to C$$

3. or for $\Box_1 = \Box_2 = \Box_3 = \Box_4 = \Box_5 = \Box_6 = \Box_8 = A$ and $\Box_7 = B$ the type

$$\alpha_3 = ((A \to A) \to A \to A) \to (A \to A) \to B \to A$$

In the following we define an algorithm which given a type-structure Θ computes a grammar(-scheme) $\mathtt{G}_\Theta$ such that for every instance α with structure Θ a context-free grammar equivalent to $\mathtt{G}_\alpha$ can be obtained from $\mathtt{G}_\Theta$ by substitution of all occurrences of indexed boxes in this grammar by the corresponding atoms in α.

Definition 5. *Let Θ be a type-structure, with indexed boxes $\Box_1, \dots, \Box_n$ and with tail $\Box_n$, i.e. $\Box_n$ is the box in the root of* $\mathtt{tree}(\Theta)$. *Let $\mathtt{G}(\Theta) = (T, N, R, S)$ be the context-free grammar with*

– *set of terminal symbols $T = \{(,), \lambda, ., x_1, \dots, x_n\}$,*
– *set of non-terminal symbols $N = \{S\} \cup \{\Box_i^P | 1 \le i \le n, P \in 2^{\{1,\dots,n\}}\}$*
– *start symbol S*

and such that R is the smallest set satisfying the following conditions.

– *if the root $\overset{|}{\Box_n}$ has as descendents primitive parts, whose roots are respectively $\Box_{i_1}, \dots, \Box_{i_t}$ in* $\mathtt{tree}(\Theta)$, *then R has exactly one production rule for S which is $S \to \lambda x_{i_1} \dots x_{i_t}.\Box_n^{\{i_1,\dots,i_t\}}$*
– *whenever a non-terminal symbol $\Box_j^P$ appears on the right side of a production rule in R, then for every $i \in P$ such that there is a (unique) primitive part in* $\mathtt{tree}(\Theta)$ *of the form $\underset{|}{\Box_i}$ there is a rule in R of the form $\Box_i^P \to x_i$*
– *whenever a non-terminal symbol $\Box_j^P$ appears on the right side of a production rule in R, then for every $i \in P$ such that p_i is of the form*

$$\begin{tikzcd} & \Box_i \arrow[dl, dash] \arrow[dr, dash] & \\ \Box_{i_1} & \dots & \Box_{i_s} \end{tikzcd}$$

and such that in $\mathtt{tree}(\Theta)$ *each $\Box_{i_l}$ has as descendents primitive parts respectively with roots $\Box_{i_l^1}, \dots, \Box_{i_l^{t_l}}$, there is a rule in R of the form*

$$\Box_i^P \to x_i(\lambda x_{i_1^1} \dots x_{i_1^{t_1}}.\Box_{i_1}^{P_1}) \dots (\lambda x_{i_s^1} \dots x_{i_s^{t_s}}.\Box_{i_s}^{P_s})$$

where $P_l = P \cup \{i_l^1, \dots, i_l^{t_l}\}$, for $1 \le l \le s$.

Example 8. Let Θ be as in example 7. Then, $\mathtt{G}_\Theta = (T, N, R, S)$ with

– $T = \{(,), \lambda, ., x_1, \dots, x_8\}$,
– $N = \{S\} \cup \{\Box_i^P \,|\, 1 \le i \le 8, P \in 2^{\{1,\dots,8\}}\}$
– start symbol S

and R given by

$$\begin{array}{llll} S & \rightarrow \lambda x_4 x_6 x_7.\ \Box_8{}^{467} & \Box_1{}^{1467} & \rightarrow x_1 \\ \Box_4{}^{467} & \rightarrow x_4(\lambda x_1.\ \Box_2{}^{1467})(\ \Box_3{}^{467}) & \Box_4{}^{1467} & \rightarrow x_4(\lambda x_1.\ \Box_2{}^{1467})(\ \Box_3{}^{1467}) \\ \Box_6{}^{467} & \rightarrow x_6(\ \Box_5{}^{467}) & \Box_6{}^{1467} & \rightarrow x_6(\ \Box_5{}^{1467}) \\ \Box_7{}^{467} & \rightarrow x_7 & \Box_7{}^{1467} & \rightarrow x_7 \end{array}$$

Naturally, for every type-scheme Θ the context-free grammar $\mathtt{G}_\Theta$ generates the empty language, since no non-terminal symbol on the right side of production rules will appear on the left side of any rule in R. But, the following proposition shows that $\mathtt{G}_\Theta$ can in fact be seen at as a scheme representing grammars for any type with structure Θ.

Proposition 5. *Let τ be a type with structure Θ, i.e. $\tau = \Theta[A_1/\Box_1, \ldots, A_n/\Box_n]$, for, not necessarily distinct, atoms $A_1, \ldots, A_n$. Let $\mathtt{G}_{\tau/\Theta}$ be the context-free grammar obtained from $\mathtt{G}_\Theta$ by substituting all occurrences of $\Box_1, \ldots, \Box_n$ respectively by $A_1, \ldots, A_n$. Then (modulo α-conversion),*

$$\begin{aligned} \mathtt{Nhabs}(\tau) &= \{M \mid \exists T \in \mathcal{L}(\mathtt{G}_\tau)\ \exists N \in \mathcal{C}(T) \textit{ such that } N \rightarrow_\eta M\} \\ &= \{M \mid \exists T \in \mathcal{L}(\mathtt{G}_{\tau/\Theta})\ \exists N \in \mathcal{C}(T) \textit{ such that } N \rightarrow_\eta M\}. \end{aligned}$$

Proof First notice that we probably use some distinct kind of enumeration for $\mathtt{G}_\tau$ and $\mathtt{G}_\Theta$ which leads to distinct indexes of variables and superscripts of non-terminal symbols. So let us consider $\mathtt{G}'_\tau$ as the grammar $\mathtt{G}_\tau$ where every index and superscript i, referring to the primitive part p_i in $\mathtt{tree}(\tau)$ is replaced by the index of the box in the root of this primitive part p_i. Since this attribution of new values to indexes is obviously injective, it is sufficient to verify that $\mathcal{L}(\mathtt{G}'_\tau) = \mathcal{L}(\mathtt{G}_\Theta)$. For this, note first that every production rule of $\mathtt{G}'_\tau$ is a production rule of $\mathtt{G}_{\tau/\Theta}$. On the other hand, all other rules in $\mathtt{G}_{\tau/\Theta}$ will never be used during the generation of a term-scheme. •

Example 9. For the types α_1, α_2 and α_3 in example 7 we obtain the following results.

1. $\mathtt{G}_{\alpha_1/\Theta}$ is obtained from $\mathtt{G}_\Theta$ substituting all occurrences of $\Box_1, \Box_2, \Box_3, \Box_5$ and $\Box_7$ by the atom A and all occurrences of $\Box_4$ and $\Box_6$ by B. The resulting set R has the following production rules

$$\begin{array}{llll} S & \rightarrow \lambda x_4 x_6 x_7.\ B^{467} & A^{1467} & \rightarrow x_1 \\ B^{467} & \rightarrow x_4(\lambda x_1.\ A^{1467})(\ A^{467}) & B^{1467} & \rightarrow x_4(\lambda x_1.\ A^{1467})(\ A^{1467}) \\ B^{467} & \rightarrow x_6(\ A^{467}) & B^{1467} & \rightarrow x_6(\ A^{1467}) \\ A^{467} & \rightarrow x_7 & A^{1467} & \rightarrow x_7 \end{array}$$

which can be simplified to

$$S \ \rightarrow\ \lambda x_4 x_6 x_7.x_4(\lambda x_1.x_1)x_7 \mid \lambda x_4 x_6 x_7.x_4(\lambda x_1.x_7)x_7 \mid \lambda x_4 x_6 x_7.x_6 x_7$$

Thus,

$$\begin{aligned} \mathtt{Nhabs}(\alpha_1) = \{\ & \lambda x_4 x_6 x_7.x_4(\lambda x_1.x_1)x_7,\ \lambda x_4 x_6 x_7.x_4(\lambda x_1.x_1), \\ & \lambda x_4 x_6 x_7.x_4(\lambda x_1.x_7)x_7,\ \lambda x_4 x_6 x_7.x_6 x_7,\ \lambda x_4 x_6.x_6\ \ \}. \end{aligned}$$

2. $\mathtt{G}_{\alpha_2/\Theta}$ has production rules

$$\begin{array}{llll}
S & \to \lambda x_4 x_6 x_7.\ C^{467} & A^{1467} & \to x_1 \\
C^{467} & \to x_4(\lambda x_1.\ A^{1467})(\ B^{467}) & C^{1467} & \to x_4(\lambda x_1.\ A^{1467})(\ B^{1467}) \\
C^{467} & \to x_6(\ C^{467}) & C^{1467} & \to x_6(\ C^{1467}) \\
B^{467} & \to x_7 & B^{1467} & \to x_7
\end{array}$$

which can be simplified to

$$S \ \to\ \lambda x_4 x_6 x_7.\ C^{467} \qquad C^{467} \ \to\ x_4(\lambda x_1.x_1)x_7 \ \mid\ x_6\ (\ C^{467})$$

Thus,

$$\begin{aligned}\mathtt{Nhabs}(\alpha_2) = \{\ & \lambda x_4 x_6 x_7.x_4(\lambda x_1.x_1)x_7, \lambda x_4 x_6.x_4(\lambda x_1.x_1), \\ & \lambda x_4 x_6 x_7.x_6(x_4(\lambda x_1.x_1)x_7), \lambda x_4 x_6 x_7.x_6(x_6(x_4(\lambda x_1.x_1)x_7)), \ldots\}\end{aligned}$$

3. $\mathtt{G}_{\alpha_3/\Theta}$ has production rules

$$\begin{array}{llll}
S & \to \lambda x_4 x_6 x_7.\ A^{467} & A^{1467} & \to x_1 \\
A^{467} & \to x_4(\lambda x_1.\ A^{1467})(\ A^{467}) & A^{1467} & \to x_4(\lambda x_1.\ A^{1467})(\ A^{1467}) \\
A^{467} & \to x_6(\ A^{467}) & A^{1467} & \to x_6(\ A^{1467}) \\
B^{467} & \to x_7 & B^{1467} & \to x_7
\end{array}$$

This grammar generates the empty language, thus $\mathtt{Nhabs}(\alpha_3) = \emptyset$.

Acknowledgements The work presented in this paper has been partially supported by funds granted to *LIACC* through the *Programa de Financiamento Plurianual, Fundação para a Ciência e Tecnologia* and *Programa POSI.*

References

1. H. Barendregt. Lambda calculi with types. In Abramsky, Gabbay, and Maibaum, eds., *Background: Computational Structures*, volume 2 of *Handbook of Logic in Computer Science*, pp. 117–309. Oxford Science Publications, 1992.
2. C.-B. Ben-Yelles. *Type-assignment in the lambda-calculus; syntax and semantics.* PhD thesis, Mathematics Dept., University of Wales Swansea, UK, 1979.
3. S. Broda and L. Damas. Counting a type's (principal) inhabitants. *Fundamenta Informaticae*, 45:33–51, 2001.
4. S. Broda, and L. Damas. On the structure of normal λ-terms having a certain type. *Proceedings of 7th WoLLIC'2000*, pp. 33–43, 2000.
5. M.W. Bunder. Proof finding algorithms for implicational logics. *Theoretical Computer Science*, 232:165–186, 2000.
6. J. R. Hindley. *Basic Simple Type Theory.* Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, 1997.
7. S. Hirokawa. Note: Infiniteness of proof(α) is polynomial-space complete. *Theoretical Computer Science*, 206:331–339, 1998.
8. W. A. Howard. The formulae-as-types notion of construction. In J. P. Seldin and J. R. Hindley, editors, *To H. B. Curry: Essays on Combinatory Logic, Lambda Calculus and Formalism*, pp. 479–490. Academic Press, 1980.
9. R. Statman. Intuitionistic propositional logic is polynomial-space complete. *Theoretical Computer Science*, 9:67–72, 1979.
10. M. Takahashi, Y. Akama, and S. Hirokawa. Normal Proofs and Their Grammar. *Information and Computation*, 125:144–153, 1996.

Permissive Belief Revision

Maria R. Cravo, João P. Cachopo, Ana C. Cachopo, and João P. Martins

Instituto Superior Técnico
Dep. Eng. Informática, GIA
Av. Rovisco Pais
1049-001 Lisboa — Portugal
{mrcravo,jcachopo,acardoso,jpm}@gia.ist.utl.pt

Abstract. We propose a new operation of belief revision, called permissive belief revision. The underlying idea of permissive belief revision is to replace the beliefs that are abandoned by traditional theories with weaker ones, entailed by them, that still keep the resulting belief set consistent. This framework allows us to keep more beliefs than what is usual using existent belief base-based revision theories.

1 Introduction

In this paper we define a new kind of belief revision. We call it permissive belief revision, and its main advantage over traditional belief revision operations is that more beliefs are kept after revising a set of beliefs. To achieve this result, permissive revision takes the beliefs abandoned by some traditional belief revision operation, and weakens them, adding their weakened versions to the result of the traditional operation. In this way, "some parts" of the abandoned beliefs are still kept.

Throughout the article we use the following notation: lower case greek letters (α, β, ...) represent meta-variables that range over single formulas; lower case roman letters (a, b, ...) represent single atomic formulas; upper case roman letters (A, B, ...) represent sets of formulas; $\mathcal{L}$ represents the language of classical logic (either propositional or first-order logic).

In Section 2 we briefly describe the work in belief revision that is relevant for the understanding of this article. In Section 3 we give some motivations, and an example, that will provide a better understanding of what is gained with permissive revision. After this, in Sections 4 and 5, we formally define this operation, and present some examples. In Section 6 we prove some properties about permissive revision, and show that it satisifies suitable counterparts for the AGM postulates. Finally, in Section 7 we discuss some relevant issues about our theory, in Section 8 we make a comparison with other approaches and in Section 9 we point out some directions in which the present work may evolve.

2 Belief Revision

One of the main sources of inspiration in belief revision, the AGM theory, follows the work of [1]. This theory deals with deductively closed sets of sentences, called sets of beliefs. According to the AGM theory, there are three operations on sets of beliefs: expansions, contractions, and revisions.

The AGM theory presents a drawback from a computational point of view, since it deals with infinite sets of beliefs. Both [8,9] and [6] modified AGM by working with a finite set of propositions, called a *belief base*, B, and using the set of consequences of B, defined as $Cn(B) = \{\phi : B \vdash \phi\}$.[1]

We also define permissive revision on finite sets of beliefs. The traditional revision of a *consistent* belief base B with a formula ϕ, represented by $(B * \phi)$, consists in changing B in such a way that it contains ϕ and is consistent (if ϕ is consistent). The case of interest is when $B \cup \{\phi\}$ is inconsistent, because, otherwise, ϕ can just be added to B.

To perform the revision $(B * \phi)$ when $B \cup \{\phi\}$ is inconsistent, we have to remove some belief(s) from B, before we can add ϕ. In other words, in a revision $(B * \phi)$ some belief(s) *must* be discarded from B.

3 Motivations

The idea of permissive revision is to transform the beliefs that were discarded in a traditional revision into weaker versions and to add them to the result of the revision. Permissive revision, thus, corresponds to a "smaller" change in beliefs than traditional revision, while keeping the goal of having a consistent result.

Conjunctions are the most obvious candidates to be weakened. This aspect was already recognized by [7], who discussed that revision theories sometimes require to give up too many beliefs, without providing a solution to the problem. While Lehmann only presents the problem regarding conjunctions, we argue that this problem is more general and that it can arise with other kinds of formulas.

To illustrate the main idea behind the weakening of conjunctions, suppose, for instance, that some traditional revision operation provides the result:

$$(\{a \wedge b, a \Rightarrow c\} * \neg c) = \{a \Rightarrow c, \neg c\}$$

Permissive revision, represented by $\circledast$, weakens the abandoned formula, $a \wedge b$ to b, and adds this to the result of traditional revision:

$$(\{a \wedge b, a \Rightarrow c\} \circledast \neg c) = \{b, a \Rightarrow c, \neg c\}$$

4 Formalization

By now, it should be clear that the main task in defining permissive revision is the definition of a function Wk, which weakens the formula that was removed during traditional revision. Actually, since there may be more than one such formula, we consider the conjunction of all the removed formulas, and weaken it into a new formula which will then be added to the result of traditional revision to obtain permissive revision.

[1] $\vdash$ represents the classical derivability operation.

The function Wk will have different definitions, depending on whether we are using classical logic, a non-monotonic logic or some other logic. In this article, we restrict ourselves to classical first order logic.

Weakening a formula depends, naturally, on the set of formulas into which we will be adding the result. Therefore, the function Wk will depend on the formula to weaken and a set of formulas:

$$\mathrm{Wk} : \mathcal{L} \times 2^{\mathcal{L}} \rightarrow \mathcal{L}$$

$\mathrm{Wk}(\phi, W)$ can be interpreted as "Weaken the formula ϕ, in such a way that after the weakened formula is added to W, the resulting set is not inconsistent".

Given such a function, we can formally define the permissive revision of a set of formulas W with a formula ϕ, $(W \circledast \phi)$. Let *Abandoned* be the conjunction of all the formulas which were abandoned during the traditional revision of W with ϕ, $\mathit{Abandoned} = \bigwedge(W - (W * \phi))$. Then, the permissive revision of W with ϕ is given by

$$(W \circledast \phi) = (W * \phi) \cup \{\mathrm{Wk}(\mathit{Abandoned}, (W * \phi))\}$$

Let us now see how a formula is weakened. Obviously, this depends on the type of formula in question. The example in the previous section conveys the main ideas behind weakening conjunctions. However, there are other logical symbols besides conjunctions. Considering the usual logical symbols, $\{\neg, \Rightarrow, \wedge, \vee, \exists, \forall\}$, we have the following definition for Wk.[2]

$$\mathrm{Wk}(\phi, W) = \begin{cases} \phi & \text{if } W \cup \{\phi\} \text{ is consistent} \\ \mathrm{WkN}(\phi, W) & \text{if } \phi \text{ is a negation} \\ \mathrm{WkI}(\phi, W) & \text{if } \phi \text{ is an implication} \\ \mathrm{WkD}(\phi, W) & \text{if } \phi \text{ is a disjunction} \\ \mathrm{WkC}(\phi, W) & \text{if } \phi \text{ is a conjunction} \\ \mathrm{WkE}(\phi, W) & \text{if } \phi \text{ is an existential rule} \\ \mathrm{WkU}(\phi, W) & \text{if } \phi \text{ is a universal rule} \\ \top & \text{otherwise} \end{cases}$$

Note that, although Wk will only be used, in the context of permissive revision, to weaken a formula ϕ known to be inconsistent with W, the weakening process is recursive (on the structure of formulas), and there may be sub-formulas which are consistent with W. That's the reason for the first case. As for the last case, which means that ϕ is an atomic formula inconsistent with W, there is no weaker formula we can give than a valid formula.

[2] This definition of Wk has some steps similar to the conversion to Conjunctive-Normal-Form, and it could be simpler if the knowledge base were required to be in a canonical form (CNF for instance). However, the syntactic differences between two logically equivalent formulas are important from the knowledge representation point of view.

Next, we define each of the weakening functions mentioned above. We should keep in mind that a good weakening function should allow us to keep as much information as possible. In order to do that for non-atomic formulas, we weaken each sub-formula and combine the results.

When $\phi = \neg\alpha$, for some atomic formula α, there is nothing we can retain of the weakening of ϕ. However, if α is a non-atomic formula, $a \vee b$, for instance, we can apply logical transformations to ϕ to bring to the surface a kind of formula we know how to handle. In this case $\neg(a \vee b)$ is logically equivalent to $(\neg a) \wedge (\neg b)$.

$$\mathrm{WkN}(\phi, W) = \begin{cases} \mathrm{Wk}(\neg\alpha \wedge \neg\beta, W) & \text{if } \phi = \neg(\alpha \vee \beta) \\ \mathrm{Wk}(\neg\alpha \vee \neg\beta, W) & \text{if } \phi = \neg(\alpha \wedge \beta) \\ \mathrm{Wk}(\alpha \wedge \neg\beta, W) & \text{if } \phi = \neg(\alpha \Rightarrow \beta) \\ \mathrm{Wk}(\alpha, W) & \text{if } \phi = \neg\neg\alpha \\ \mathrm{Wk}(\forall(x)\neg\alpha(x), W) & \text{if } \phi = \neg\exists(x)\alpha(x) \\ \mathrm{Wk}(\exists(x)\neg\alpha(x), W) & \text{if } \phi = \neg\forall(x)\alpha(x) \\ \top & \text{otherwise} \end{cases}$$

Weakening an implication is treated in a similar way, transforming the implication into the logically equivalent disjunction, and weakening the result instead.

$$\mathrm{WkI}(\alpha \Rightarrow \beta, W) = \mathrm{Wk}(\neg\alpha \vee \beta, W)$$

If $\phi = \alpha \vee \beta$, and it is inconsistent with W (otherwise WkD would not be used), then both α and β are inconsistent with W. So, to weaken ϕ we have to individually weaken both α and β, in W, and combine the results with the disjunction again.

$$\mathrm{WkD}(\alpha \vee \beta, W) = \mathrm{Wk}(\alpha, W) \vee \mathrm{Wk}(\beta, W)$$

Conjunction seems to be a more complex case. To help understand its definition we present some examples. First, consider the set $W = \{a \wedge b\}$ and its revision with $\neg a$. Using permissive revision, we use $\mathrm{Wk}(a \wedge b, \{\neg a\})$ and expect it to give b. We just have to abandon one of the elements of the conjunction and keep the other. However, if each element is itself a non-atomic formula, the contradiction may be deeper inside in either one or in both of the elements of the conjunction. For instance, given $W = \{(a \wedge b) \wedge (c \wedge d)\}$ and revising it with $\neg(b \wedge c)$ we would like to get $(a \wedge (c \wedge d)) \vee ((a \wedge b) \wedge d)$, i.e., if it's not possible to have both b and c, then we would like to have either a, b and d or a, c and d. This is the result of $\mathrm{WkC}((a \wedge b) \wedge (c \wedge d), \{\neg(b \wedge c)\})$, according to the following definition.

$$\begin{aligned} \mathrm{WkC}(\alpha \wedge \beta, W) = {} & (\mathrm{Wk}(\alpha, W) \wedge \mathrm{Wk}(\beta, W \cup \{\mathrm{Wk}(\alpha, W)\})) \vee \\ & (\mathrm{Wk}(\beta, W) \wedge \mathrm{Wk}(\alpha, W \cup \{\mathrm{Wk}(\beta, W)\})) \end{aligned}$$

Handling existentially quantified formulas will be done through skolemization, weakening the formula which results from the elimination of the existential quantifier.

$$\mathrm{WkE}(\exists(x)\alpha(x), W) = \mathrm{Wk}(\alpha(p), W), \quad \text{where } p \text{ is a Skolem constant}$$

Finally, the result of weakening universally quantified formulas is just $\top$. This means that, in what concerns this kind of formula, permissive revision brings nothing new. In Section 9, we discuss some alternatives to the weakening of universally quantified formulas.

$$\mathrm{WkU}(\forall(x)\alpha(x), W) = \top$$

5 Examples

In this section we present some examples, to illustrate permissive revision. In all the examples we present, permissive revision keeps more beliefs than traditional revision. Of course, this is not always the case. Sometimes both revisions give the same result.

Example 1 (Weakening of conjunctions). In the first situation both conjuncts are inconsistent with the result of traditional revision; in the second situation only one of the conjuncts is inconsistent; and in the third situation none of the conjuncts by itself is inconsistent, only the conjunction of them causes the inconsistency.

1. **Both conjuncts are inconsistent**

$$W = \{a \wedge (b \wedge c), a \Rightarrow d, b \Rightarrow d\}$$

suppose

$$(W * \neg d) = \{a \Rightarrow d, b \Rightarrow d, \neg d\}$$

then

$$\begin{aligned}
&\mathrm{Wk}(a \wedge (b \wedge c), (W * \neg d)) = \\
&\quad = (\mathrm{Wk}(a, (W * \neg d)) \wedge \mathrm{Wk}(b \wedge c, (W * \neg d) \cup \{\mathrm{Wk}(a, (W * \neg d))\})) \vee \\
&\quad\quad (\mathrm{Wk}(b \wedge c, (W * \neg d)) \wedge \mathrm{Wk}(a, (W * \neg d) \cup \{\mathrm{Wk}(b \wedge c, (W * \neg d))\})) \\
&\quad = (\top \wedge \mathrm{Wk}(b \wedge c, (W * \neg d))) \vee (\mathrm{Wk}(b \wedge c, (W * \neg d)) \wedge \top) \\
&\quad = \mathrm{Wk}(b \wedge c, (W * \neg d)) \\
&\quad = (\mathrm{Wk}(b, (W * \neg d)) \wedge \mathrm{Wk}(c, (W * \neg d) \cup \{\mathrm{Wk}(b, (W * \neg d))\})) \vee \\
&\quad\quad (\mathrm{Wk}(c, (W * \neg d)) \wedge \mathrm{Wk}(b, (W * \neg d) \cup \{\mathrm{Wk}(c, (W * \neg d))\})) \\
&\quad = (\top \wedge c) \vee (c \wedge \top) \\
&\quad = c
\end{aligned}$$

and

$$(W \circledast \neg d) = \{a \Rightarrow d, b \Rightarrow d, \neg d, c\}$$

Note that in the traditional revision we can no longer derive c, but this is still a consequence of the permissive revision.

2. **Only one of the conjuncts is inconsistent**

$$W = \{a \wedge b, a \Rightarrow c\}$$

suppose

$$(W * \neg c) = \{a \Rightarrow c, \neg c\}$$

then

$$\begin{aligned}
&\mathrm{Wk}(a \wedge b, (W * \neg c)) = \\
&\quad = (\mathrm{Wk}(a, (W * \neg c)) \wedge \mathrm{Wk}(b, (W * \neg c) \cup \{\mathrm{Wk}(a, (W * \neg c))\})) \vee \\
&\qquad (\mathrm{Wk}(b, (W * \neg c)) \wedge \mathrm{Wk}(a, (W * \neg c) \cup \{\mathrm{Wk}(b, (W * \neg c))\})) \\
&\quad = (\top \wedge \mathrm{Wk}(b, (W * \neg c) \cup \{\top\})) \vee (b \wedge \mathrm{Wk}(a, (W * \neg c) \cup \{b\})) \\
&\quad = (\top \wedge b) \vee (b \wedge \top) \\
&\quad = b
\end{aligned}$$

and

$$(W \circledast \neg c) = \{a \Rightarrow c, \neg c, b\}$$

Like before, we keep more beliefs than traditional revision, namely b.

3. **None of the conjuncts by itself is inconsistent**

$$W = \{a \wedge b, (a \wedge b) \Rightarrow c\}$$

suppose

$$(W * \neg c) = \{(a \wedge b) \Rightarrow c, \neg c\}$$

then

$$\begin{aligned}
&\mathrm{Wk}(a \wedge b, (W * \neg c)) = \\
&\quad = (\mathrm{Wk}(a, (W * \neg c)) \wedge \mathrm{Wk}(b, (W * \neg c) \cup \{\mathrm{Wk}(a, (W * \neg c))\})) \vee \\
&\qquad (\mathrm{Wk}(b, (W * \neg c)) \wedge \mathrm{Wk}(a, (W * \neg c) \cup \{\mathrm{Wk}(b, (W * \neg c))\})) \\
&\quad = (a \wedge \mathrm{Wk}(b, (W * \neg c) \cup \{a\})) \vee (b \wedge \mathrm{Wk}(a, (W * \neg c) \cup \{b\})) \\
&\quad = (a \wedge \top) \vee (b \wedge \top) \\
&\quad = a \vee b
\end{aligned}$$

and

$$(W \circledast \neg c) = \{(a \wedge b) \Rightarrow c, \neg c, a \vee b\}$$

Example 2 (Weakening of disjunctions). We now present one example of weakening a disjunction. Obviously, both disjuncts are inconsistent with the result of traditional revision, otherwise the disjunction would not be inconsistent. Furthermore, the disjuncts are both non-atomic, otherwise the result would be $\top$.

$$W = \{(a \wedge b) \vee (c \wedge d), b \Rightarrow e, d \Rightarrow e, a \Rightarrow f, c \Rightarrow f\}$$

suppose

$$(W * \neg e) = \{b \Rightarrow e, d \Rightarrow e, a \Rightarrow f, c \Rightarrow f, \neg e\}$$

then

$$\begin{aligned}
&\mathrm{Wk}((a \wedge b) \vee (c \wedge d), (W * \neg e)) = \\
&\quad = \mathrm{WkD}((a \wedge b) \vee (c \wedge d), (W * \neg e)) \\
&\quad = \mathrm{Wk}(a \wedge b, (W * \neg e)) \vee \mathrm{Wk}(c \wedge d, (W * \neg e)) \\
&\quad = \mathrm{WkC}(a \wedge b, (W * \neg e)) \vee \mathrm{WkC}(c \wedge d, (W * \neg e)) \\
&\quad = a \vee c
\end{aligned}$$

and

$$(W \circledast \neg e) = \{a \vee c, b \Rightarrow e, d \Rightarrow e, a \Rightarrow f, c \Rightarrow f, \neg e\}$$

Note that in the traditional revision we can no longer derive, for instance f, but this is still a consequence of the permissive revision.

Example 3 (Weakening an existentially quantified formula).

$$W = \{\exists(x)a(x) \wedge b(x), \forall(x)a(x) \Rightarrow c(x)\}$$

suppose

$$(W * \forall(x)\neg c(x)) = \{\forall(x)a(x) \Rightarrow c(x), \forall(x)\neg c(x)\}$$

then

$$\begin{aligned}
&\mathrm{Wk}(\exists(x)a(x) \wedge b(x), (W * \forall(x)\neg c(x))) = \\
&\quad = \mathrm{WkE}(\exists(x)a(x) \wedge b(x), (W * \forall(x)\neg c(x))) \\
&\quad = \mathrm{Wk}(a(p) \wedge b(p), (W * \forall(x)\neg c(x))) \\
&\quad = \mathrm{WkC}(a(p) \wedge b(p), (W * \forall(x)\neg c(x))) \\
&\quad = b(p)
\end{aligned}$$

where p is a Skolem constant and

$$(W \circledast \forall(x)\neg c(x)) = \{\forall(x)a(x) \Rightarrow c(x), \forall(x)\neg c(x), b(p)\}$$

In words, permissive revision, unlike traditional revision, allows us to keep the belief $\exists(x)b(x)$.

6 Properties

We now prove two essential properties of the Wk function. By essential properties, we mean that it would be unacceptable for the Wk function not to satisfy them. The first property ensures that we don't produce an inconsistent set when we add the result of weakening a formula to the result of the traditional revision. The second property ensures that we are not able to derive new conclusions from the result of weakening a formula, that were not derivable from the formula itself.

The next theorem guarantees the first of these properties.

Theorem 1 *Let W be a consistent set of formulas, and ϕ any formula. Then $W \cup \{\mathrm{Wk}(\phi, W)\}$ is consistent.*

Proof. If ϕ is consistent with W, then $\mathrm{Wk}(\phi, W) = \phi$ and the result follows trivially. Otherwise, we will prove by induction on the structure of the formula ϕ that the weakening function produces a formula consistent with W.

If ϕ is a literal (an atomic formula or the negation of an atomic formula) or a universally quantified formula, then $\mathrm{Wk}(\phi, W) = \top$, and therefore $W \cup \{\top\}$ is consistent, provided that W is consistent.

The cases where ϕ is of the form $\neg\alpha$ or $\alpha \Rightarrow \beta$, reduce to one of the other cases, since the weakening of ϕ in these cases reduces to the weakening of a logically equivalent formula, with either a quantifier, a disjunction or a conjunction.

Assume that α, β and $\gamma(p)$, where p is some constant, are formulas that verify the theorem. Since $W \cup \{\mathrm{Wk}(\gamma(p), W)\}$ is consistent by hypothesis, then $W \cup \{\mathrm{Wk}(\exists(x)\gamma(x), W)\}$ is also consistent, by definition of WkE. Accordingly, given that $W \cup \{\mathrm{Wk}(\alpha, W)\}$ is consistent, and, therefore, $W \cup \{\mathrm{Wk}(\alpha, W) \vee \mathrm{Wk}(\beta, W)\}$ is consistent, we prove that $W \cup \{\mathrm{Wk}(\alpha \vee \beta, W)\}$ is also consistent. Finally, let $W' = W \cup \{\mathrm{Wk}(\alpha, W)\}$, which, as we have seen, is consistent. Since, by hypothesis, $W' \cup \{\mathrm{Wk}(\beta, W')\}$ is consistent, i.e., $W \cup \{\mathrm{Wk}(\alpha, W), \mathrm{Wk}(\beta, W')\}$ is consistent, we have that $W \cup \{\mathrm{Wk}(\alpha, W) \wedge \mathrm{Wk}(\beta, W')\}$ is consistent, from where it follows trivially that $W \cup \{\mathrm{Wk}(\alpha \wedge \beta, W)\}$ is consistent, which finishes our proof. □

Theorem 2 guarantees that the result of weakening a formula is not stronger than the original formula, i.e., we do not introduce new beliefs.

Theorem 2 *Let W be a set of formulas, and ϕ any formula. Then $\phi \vdash \mathrm{Wk}(\phi, W)$.*

Proof. If $\phi \vdash \bot$ then $\phi \vdash \psi$ for every formula ψ, and in particular for $\psi = \mathrm{Wk}(\phi, W)$. If ϕ is consistent with W, then $\mathrm{Wk}(\phi, W) = \phi$ and, obviously, $\phi \vdash \phi = \mathrm{Wk}(\phi, W)$. Otherwise, as above, we will prove by induction on the structure of the formula ϕ that the weakening function produces a formula not stronger than the original.

The structure of this proof is similar to the previous one: if ϕ is a literal or a universally quantified formula, then $\mathrm{Wk}(\phi, W) = \top$, and $\phi \vdash \top$; if ϕ is of the form $\neg\alpha$ or $\alpha \Rightarrow \beta$, the weakening of ϕ reduces to the weakening of a logical equivalent formula, with either a quantifier, a disjunction or a conjunction.

By eliminating the existential quantifier, we have that $\exists(x)\gamma(x) \vdash \gamma(p)$ for some Skolem constant p. By hypothesis, $\gamma(p) \vdash \mathrm{Wk}(\gamma(p), W) = \mathrm{Wk}(\exists(x)\gamma(x), W)$, and, therefore, $\exists(x)\gamma(x) \vdash \mathrm{Wk}(\exists(x)\gamma(x), W)$.

Assume that α and β are formulas that verify the theorem. Given that, by hypothesis, $\alpha \vdash \mathrm{Wk}(\alpha, W)$, then $\alpha \vdash \mathrm{Wk}(\alpha, W) \vee \mathrm{Wk}(\beta, W)$, and, likewise, since $\beta \vdash \mathrm{Wk}(\beta, W)$ then $\beta \vdash \mathrm{Wk}(\alpha, W) \vee \mathrm{Wk}(\beta, W)$. Joining the two, we have that $\alpha \vee \beta \vdash \mathrm{Wk}(\alpha, W) \vee \mathrm{Wk}(\beta, W)$, i.e., $\alpha \vee \beta \vdash \mathrm{Wk}(\alpha \vee \beta, W)$. To finish the proof, let's see that conjunction preserves the theorem: from $\alpha \vdash \mathrm{Wk}(\alpha, W)$ and $\beta \vdash \mathrm{Wk}(\beta, W \cup \{\mathrm{Wk}(\alpha, W)\})$, it follows trivially that $\alpha \wedge \beta \vdash \mathrm{Wk}(\alpha \wedge \beta, W)$. $\square$

Although we don't consider it essential, we now prove another theorem that will be needed when we prove the satisfaction of the AGM postulates for our theory. The theorem says that the results of weakening a formula, with respect to two logically equivalent sets, are the same.

Theorem 3 *Let W and W' be two sets of formulas, such that $Cn(W) = Cn(W')$, and ϕ any formula. Then* $\mathrm{Wk}(\phi, W) = \mathrm{Wk}(\phi, W')$.

Proof. The only use of W in the definition of Wk is to check whether $W \cup \{\phi\}$ is consistent. Since $Cn(W) = Cn(W')$, $W \cup \{\phi\}$ is consistent, iff $W' \cup \{\phi\}$ is consistent. $\square$

We now show that permissive revision satisfies the AGM postulates for revision, if the traditional revision satisfies these postulates. Since these postulates refer to a revision operation on belief sets (theories or closed sets), and permissive revision was defined on bases (finite sets), the first thing to do is to define a corresponding permissive revision on theories.

Let W be a base, T the theory generated by W, i.e., $T = Cn(W)$, and ϕ a formula. The permissive revision of the theory T with the formula ϕ, $(T \circledast_T \phi)$, is defined by

$$(T \circledast_T \phi) = Cn(W \circledast \phi)$$

Note that this definition implies that $(T \circledast_T \phi)$ depends not only on T and ϕ, but also on the base W from which T was generated.

We recall that permissive revision is defined in terms of a traditional revision, $*$, by:

$$(W \circledast \phi) = (W * \phi) \cup \{\mathrm{Wk}(\mathit{Abandoned}, (W * \phi))\}$$

Before we prove anything on permissive revision, let us assume that the traditional revision on bases, $*$, satisfies suitable counterparts to the AGM postulates.

$$
\begin{array}{ll}
(*1) & (W * \phi) \text{ is a base} \\
(*2) & \phi \in (W * \phi) \\
(*3) & (W * \phi) \subseteq W \cup \{\phi\} \\
(*4) & \text{If } \neg\phi \notin Cn(W), \text{then } W \cup \{\phi\} \subseteq (W * \phi) \\
(*5) & (W * \phi) \text{ is inconsistent, iff } \neg\phi \in Cn(\emptyset) \\
(*6) & \text{If } \phi \Leftrightarrow \psi \in Cn(\emptyset), \text{then } (W * \phi) - \{\phi\} = (W * \psi) - \{\psi\}
\end{array}
$$

Of these postulates, only postulate (∗6) is not a straightforward counterpart to the corresponding AGM postulate. The straightforward counterpart would be

$$\text{If } \phi \Leftrightarrow \psi \in Cn(\emptyset), \text{ then } (W * \phi) = (W * \psi)$$

Since we are dealing with bases, and not closed sets, it is not reasonable to expect such a result (unless, of course, ϕ and ψ are not only equivalent, but also the same formula). What is reasonable to assume is that the revisions of the same base with two equivalent (but different) formulas, only differ between them in these formulas.

If we now define a traditional revision on theories as

$$(T *_T \phi) = Cn(W * \phi)$$

where T is the theory generated by the base W, $T = Cn(W)$, it is trivial to show that the AGM postulates are satisfied by $*_T$.

$$
\begin{array}{ll}
(*_T1) & (T *_T \phi) \text{ is a theory, i.e., } (T *_T \phi) = Cn(T *_T \phi) \\
(*_T2) & \phi \in (T *_T \phi) \\
(*_T3) & (T *_T \phi) \subseteq Cn(T \cup \{\phi\}) \\
(*_T4) & \text{If } \neg\phi \notin T, \text{then } Cn(T \cup \{\phi\}) \subseteq (T *_T \phi) \\
(*_T5) & (T *_T \phi) \text{ is inconsistent iff } \neg\phi \in Cn(\emptyset) \\
(*_T6) & \text{If } \phi \Leftrightarrow \psi \in Cn(\emptyset), \text{then } (T *_T \phi) = (T *_T \psi)
\end{array}
$$

Now that we have established the postulates satisfied by the traditional revision on which permissive revision is based, we prove the following theorem.

Theorem 4 *Let $\circledast_T$ be a permissive revision on theories defined as before:*

$$(T \circledast_T \phi) = Cn(W \circledast \phi).$$

Then, for any theory T (generated from a base W), and any formulas ϕ and ψ, $\circledast_T$ satisfies the AGM postulates.

$$
\begin{array}{ll}
(\circledast_T1) & (T \circledast_T \phi) \textit{ is a theory, i.e., } (T \circledast_T \phi) = Cn(T *_T \phi) \\
(\circledast_T2) & \phi \in (T \circledast_T \phi) \\
(\circledast_T3) & (T \circledast_T \phi) \subseteq Cn(T \cup \{\phi\}) \\
(\circledast_T4) & \textit{If } \neg\phi \notin T, \textit{ then } Cn(T \cup \{\phi\}) \subseteq (T \circledast_T \phi) \\
(\circledast_T5) & (T \circledast_T \phi) \textit{ is inconsistent, iff } \neg\phi \in Cn(\emptyset) \\
(\circledast_T6) & \textit{If } \phi \Leftrightarrow \psi \in Cn(\emptyset), \textit{ then } (T \circledast_T \phi) = (T \circledast_T \psi)
\end{array}
$$

Proof.
($\circledast_T 1$) $(T \circledast_T \phi)$ is a theory, i.e., $(T \circledast_T \phi) = Cn(T *_T \phi)$.
By definition of $\circledast_T$.

($\circledast_T 2$) $\phi \in (T \circledast_T \phi)$.
By definition of $\circledast_T$ and ($*2$).

($\circledast_T 3$) $(T \circledast_T \phi) \subseteq Cn(T \cup \{\phi\})$.
By definition of $\circledast_T$ and $\circledast$

$$(T \circledast_T \phi) = Cn(W \circledast \phi) = Cn((W * \phi) \cup \{\text{Wk}(\mathit{Abandoned}, (W * \phi))\})$$

By ($*3$) ($(W * \phi) \subseteq W \cup \{\phi\}$), Theorem 2 ($\mathit{Abandoned} \vdash \text{Wk}(\mathit{Abandoned}, W)$), and monotonicity

$$Cn((W * \phi) \cup \{\text{Wk}(\mathit{Abandoned}, (W * \phi))\}) \subseteq Cn(W \cup \{\phi\} \cup \{\mathit{Abandoned}\})$$

By definition of *Abandoned*, $\mathit{Abandoned} = \bigwedge(W - (W * \phi))$, we have that $W \vdash \mathit{Abandoned}$, so

$$Cn(W \cup \{\phi\} \cup \{\mathit{Abandoned}\}) = Cn(W \cup \{\phi\})$$

Finally, since

$$Cn(W \cup \{\phi\}) = Cn(Cn(W) \cup \{\phi\}) = Cn(T \cup \{\phi\})$$

our proof of ($\circledast_T 3$) is complete.

($\circledast_T 4$) If $\neg\phi \notin T$, then $Cn(T \cup \{\phi\}) \subseteq (T \circledast_T \phi)$.

$$Cn(T \cup \{\phi\}) = Cn(Cn(W) \cup \{\phi\}) = Cn(W \cup \{\phi\})$$

By ($*4$), since $\neg\phi \notin T$, we have that $W \cup \{\phi\} \subseteq (W * \phi)$. This, together with monotonicity, implies that

$$Cn(W \cup \{\phi\}) \subseteq Cn(W * \phi)$$

Now, if $\neg\phi \notin T$, then, by definition, $\mathit{Abandoned} = \bigwedge\{\} = \top$, and $Wk(\mathit{Abandoned}, (W * \phi)) = \top$. So, $(W \circledast \phi) = (W * \phi)$, and $(T \circledast_T \phi) = Cn(W * \phi)$.

($\circledast_T 5$) $(T \circledast_T \phi)$ is inconsistent, iff $\neg\phi \in Cn(\emptyset)$.
By definition,

$$(T \circledast_T \phi) = Cn(W \circledast \phi) = Cn((W * \phi) \cup \{\text{Wk}(\mathit{Abandoned}, (W * \phi))\}).$$

If $\neg\phi \in Cn(\emptyset)$, then, by ($*5$), $(W * \phi)$ is inconsistent, and so is $(T \circledast_T \phi)$.
If $\neg\phi \notin Cn(\emptyset)$, then, by ($*5$), $(W * \phi)$ is consistent, and, by Theorem 1 so is $(W * \phi) \cup \{\text{Wk}(\mathit{Abandoned}, (W * \phi))\}$.

($\circledast_T 6$) If $\phi \Leftrightarrow \psi \in Cn(\emptyset)$, then $(T \circledast_T \phi) = (T \circledast_T \psi)$.
By definition,

$$(T \circledast_T \phi) = Cn(W \circledast \phi) = Cn((W * \phi) \cup \{\mathrm{Wk}(\mathit{Abandoned}_\phi, (W * \phi))\}),$$

where $\mathit{Abandoned}_\phi = \bigwedge(W - (W * \phi))$, and

$$(T \circledast_T \psi) = Cn(W \circledast \psi) = Cn((W * \psi) \cup \{\mathrm{Wk}(\mathit{Abandoned}_\psi, (W * \psi))\}),$$

where $\mathit{Abandoned}_\psi = \bigwedge(W - (W * \psi))$.
Since, by ($*6$), $(W * \phi) - \{\phi\} = (W * \psi) - \{\psi\}$, we have that

$$\mathit{Abandoned}_\phi = \mathit{Abandoned}_\psi$$

Since, by ($*_T 6$), $Cn(W * \phi) = Cn(W * \psi)$, by Theorem 3, we have that

$$\mathrm{Wk}(\mathit{Abandoned}_\phi, (W * \phi)) = \mathrm{Wk}(\mathit{Abandoned}_\psi, (W * \psi))$$

which ends our proof. □

7 Discussion

Traditional belief revision theories [9,6] may produce different results when revising logically equivalent theories with the same formula, i.e., they are syntax-dependent. For example, the fact that both a and b are true may be represented either by $\{a \wedge b\}$ or by $\{a, b\}$. These two representations will provide different results when revised with $\neg a$. This should be expected, since we are dealing with syntax-based approaches and finite belief sets. However, *in this example*, if permissive revision is applied to weaken the formulas removed by traditional revision, the result will be the same, $\{b\}$. This allows us to conclude that, in some cases, the syntax-dependency of traditional approaches is nullified by permissive revision. Furthermore, the weakening function by itself is syntax-dependent, as the following example shows: $\mathrm{Wk}(a \vee b \Rightarrow c, \{a, \neg c\}) = \top$, but $\mathrm{Wk}((a \Rightarrow c) \wedge (b \Rightarrow c), \{a, \neg c\}) = b \Rightarrow c$. Again, such a behaviour should be expected: the weakening function completely relies on the syntax of the formula to be weakened. If this dependency was considered a flaw, the use of a canonical form for the formula to be weakened would very easily eliminate it. However, since the weakening function is applied to the result of a traditional theory, which is syntax-dependent, it wouldn't make much sense. Theorem 3, on the other hand, shows that this function is *not* dependent on the syntax of the set in respect to which a formula is weakened.

A preliminary report of the work presented in this article appears in [3]. The present article contains, in addition to the preliminary report, Theorem 3 and the proof of the AGM postulates.

8 Comparison with Other Approaches

To the best of our knowledge, when we submitted this article no similar work had been done, except for our preliminary report [3]. The work presented in [2] however, also aims at minimizing the loss of information by weakening information involved in conflicts rather than completely removing it. We will first convey the main ideas behind their work, and then present some comments on the comparison of both approaches.

In [2] it is assumed that the available information is given as an ordered knowledge base (KB), i.e., a ranking of information as logical sentences: $KB = \{S_1, S_2, \ldots, S_n\}$. When revising a KB with a formula ϕ, they start with $i = 1$ and $KB = \{\phi\}$; for each S_i, if it is consistent with KB then $KB \leftarrow KB \cup S_i$. Otherwise, all possible disjunctions (of the formulas in conflict) of size 2 are computed. If they are consistent with KB then they are added to KB. Otherwise, all possible disjunctions of size 3 are computed, and so on.

One major difference between both approaches is that [2] is a "complete" revision operation, while ours can be applied to the result of any traditional revision operation.[3] So, our theory is also more permissive in the sense that it allows any traditional theory to choose the formulas to weaken.

The only example in [2] has its knowledge base in clausal form, although this does not seem to be a requirement. If we convert the examples in this paper to clausal form, both approaches produce exactly the same results in all the examples. Further work is needed to prove whether this is always so. However, if we use our original examples, the results are not same. Actually, since in all our examples only one formula is removed, the work of [2] simply discards that formula, while ours weakens it.

9 Future Work

As we saw in Section 4, universal rules are weakened to $\top$, which is obviously too drastic a solution. This aspect can be improved in two directions. When considering a monotonic logic, a universal rule can be weakened following the general ideas presented in Section 4. For instance, if we have $\forall(x)a(x) \wedge b(x)$, and revise this with $\neg a(p)$, the universal rule must be abandoned, but it can be weakened to $\forall(x)b(x)$.

In another direction, i.e., when considering a non-monotonic logic, the most natural way of weakening a universal rule is to turn it into the "corresponding" default rule. Of course, defining the exact meaning of "corresponding" default rule will depend on the particular non-monotonic logic being considered, but we can state this informally as turning a universal like "All As are Bs" into the default "Typically, As are Bs". See [10] for an approach to this problem, using Default Logic.

We intend to implement permissive revision on top of SNePSwD [4]. The fact that this system is a belief revision system, with an underlying non-monotonic

[3] Our approach could even be applied to the result of theirs.

logic [5] will be particularly helpful. This system already has mechanisms for determining the consistency of belief sets, and keeping a record of inconsistent sets, which will be necessary for the implementation of the weakening function.

10 Acknowledgements

This work was partially supported by IDMEC—Institute of Mechanical Engineering, Intelligent Systems Centre.

References

1. Carlos E. Alchourr0n, Peter G rdenfors, and David Makinson. On the logic of theory change: partial meet functions for contraction and revision. *The Journal of Symbolic Logic*, 50(2):510–530, 1985.
2. Salem Benferhat, Souhila Kaci, Daniel Le Berre, and Mary-Anne Williams. Weakening conflicting information for iterated revision and knowledge integration. In *Proceedings of IJCAI-2001*, Seattle, Washington, USA, 2001. Morgan Kaufmann Publishers, Inc.
3. Maria R. Cravo, Jo o P. Cachopo, Ana C. Cachopo, and Jo o P. Martins. Permissive belief revision (preliminary report). "Third Workshop on Nonmonotonic Reasoning, Action, and Change" (NRAC99), "Sixteenth International Joint Conference on Artificial Intelligence" (IJCAI99), August 1999.
4. Maria R. Cravo and Jo o P. Martins. SNePSwD, a newcomer to the SNePS family. *Journal of Experimental and Theoretical Artificial Intelligence*, 5:135–148, 1993.
5. Maria R. Cravo. SWMC: A logic for default reasoning and belief revision (a new version). Technical Report GIA 93/02, Instituto Superior T cnico, Universidade T cnica de Lisboa, Lisbon, Portugal, November 1993.
6. Andr Fuhrmann. Theory contraction through base contraction. *Journal of Philosophical Logic*, 20(2):175–203, 1991.
7. Daniel Lehmann. Belief revision revised. In *Proceedings of IJCAI-95*, pages 1534–1540, Montreal, Canada, 1995. Morgan Kaufmann Publishers, Inc.
8. Bernhard Nebel. A knowledge level analysis of belief revision. In *Proceedings of KR-89*, pages 301–311. Morgan Kaufmann Publishers, Inc., Toronto, Canada, 1989.
9. Bernhard Nebel. *Reasoning and revision in hybrid representation systems*, volume 422 of *Lecture Notes in Artificial Intelligence*. Springer-Verlag, Heidelberg, Germany, 1990.
10. C.F. Pimentel and M. R. Cravo. Permissive belief revision using default logic. Technical Report GIA 01/01, Instituto Superior T cnico, Universidade T cnica de Lisboa, Lisbon, Portugal, 2001.

Global Hull Consistency with Local Search for Continuous Constraint Solving

Jorge Cruz and Pedro Barahona

Dep. de Informática, Universidade Nova de Lisboa, 2829-516 Caparica, Portugal
{jc,pb}@di.fct.unl.pt

Abstract. This paper addresses constraint solving over continuous domains in the context of decision making, and discusses the trade-off between precision in the definition of the solution space and the computational effort required. In alternative to local consistency, which is usually maintained in handling continuous constraints, we discuss maintaining global hull-consistency. Experimental results show that this may be an appropriate choice, achieving acceptable precision with relatively low computational cost. The approach relies on efficient algorithms and the best results are obtained with the integration of a local search procedure within interval constraint propagation.

1 Introduction

Model-based decision support relies on an explicit representation of a system in some domain of interest. Given the inevitable simplification that a model introduces, rather than obtaining the solution in the (simplified) model that optimises a certain goal, a decisor is often more interested in finding a compact representation of the solutions that satisfy some constraints. From the extra knowledge that the decisor might have, the set of acceptable solutions may be subsequently focussed into some region of interest defined by means of additional constraints.

An interesting implementation of these ideas, is presented in [8] which addresses this type of problems in the domain of engineering. In this domain, models are usually specified as a set of constraints on continuous variables (ranging over the reals). Moreover, these constraints are usually non linear, which makes their processing quite difficult, since any small errors may easily be expanded throughout computation. The approach is based on the exploitation of octrees in order to determine the regions of the 3D space that satisfy the intended constraints. Although the solution proposed is rather ingenuous, its application to problems with non-convex solution spaces runs into difficulties, namely the large number of convex solutions that might be computed.

Rather than finding precise bounds of many convex 3D spaces, a simpler approach relies on defining the upper and lower bounds of every variable that contains the solution space. Although less precise than the former, subsequent interaction with the user may shorten these bounds to specific regions of interest. This approach relies on reasoning about variables that lie on certain intervals, namely the ability to shorten their interval domain by pruning the (outward) regions where it is possible to prove that no solutions may exist. This is the aim of interval constraints, where variables range over intervals and are subject to constraint propagation to reduce their domains.

As with other domains, constraint propagation of intervals maintains some form of local consistency of the constraint set. Typically, two types of local consistency are considered: box-consistency [1, 15] and hull-consistency [9, 2]. Neither of them is complete and the pruning of the domains that is obtained is often quite poor. To improve this pruning maintenance of higher order local consistencies (3B-consistency [9], Bound-consistency [13]) might be envisaged, but their implementation is complex and published results are very scarse. We have recently proposed global hull-consistency [4, 6], but its implementation suffered from similar efficiency difficulties.

In this paper, we present improved procedures that maintain global hull-consistency, and show that the best results are obtained with the integration of constraint propagation with a form of local search commonly adopted in multidimensional root finding over the reals.

The paper is organised as follows. To make the paper self contained, sections 2 and 3 overview the main concepts involved in, respectively, constraint solving in continuous domains and interval constraints. Section 4 presents the definition of global hull-consistency and describes several procedures to enforce it. One such procedure requires the search for a solution, and a local search implementation of such procedure is presented in section 5. Section 6 compares the results obtained on a particular example, by a) applying the octree approach; b) maintaining local consistency and c) maintaining global-hull consistency. For the latter case, the section discusses the efficiency of the different procedures. Finally, the main conclusions are summarised and future research directions discussed.

2 Continuous Constraint Solving Problems

Many problems of the real world can be modelled as constraint satisfaction problems (CSPs). A CSP is defined by a set of variables each with an associated domain of possible values and a set of constraints on subsets of the variables. A constraint specifies which values from the domains of its variables are compatible. A solution is an assignment of values to all variables, which satisfies all the constraints.

The notion of CSP was initially introduced [10] to address combinatorial problems over finite domains. Continuous CSPs (CCSPs) [8] are extensions of the earlier CSP framework to address variables with continuous domains. In CCSPs all variable domains are continuous real intervals and the constraints are specified as numeric relations, either equalities or inequalities.

The CCSP framework is powerful enough to model a wide range of real world problems, in particular problems involving uncertain continuous parameters. A CCSP can have one, several or no solutions. In many practical applications the modelling of a problem as a CCSP is embedded in a larger decision process. Depending on this decision process it may be desirable to determine whether a solution exists (verify the consistency of the CCSP), to find one solution, to compute the space of all solutions of the CCSP, or to find an optimal solution relative to a given cost function.

In this work we address under-constrained CCSPs, that is, CCSPs with a large (usually infinite) solution set, where the goal is not to find a particular solution but rather to capture the set of all solutions. This is often the case of continuous decision problems where the constraints are not enough to identify a single solution but may circumscribe the set of possibilities to some specific regions of the search space.

2.1 The Representation of Continuous Domains

In CCSPs, the initial domains associated with the variables are infinite connected sets of real numbers called real intervals. A real interval, which can be closed, open or half open, will be generally denoted by $<a..b>$ where a and b represent the bounds.

In practice, computer systems are restricted to represent a finite subset of the real numbers, the floating-point numbers. Several authors [9, 1, 15] have defined the set F of machine numbers (F-numbers) as the set of floating-point numbers augmented with the two infinity symbols ($-\infty$ and $+\infty$). F is totally ordered and if f is an F-number f^- and f^+ denote respectively the two F-numbers immediately below and immediately above f in the total order. An F-interval is a closed real interval $[a..b]$ where a and b are F-numbers. In particular, if $b=a$ or $b=a^+$ the F-interval is called *canonical*.

The finite subset of the real intervals that can be represented by a particular machine is the set of all F-intervals. However, any real interval may be associated with a larger F-interval (wrt set inclusion). The F-interval approximation of a real interval $<a..b>$ is the smallest F-interval $[\lfloor a \rfloor..\lceil b \rceil]$ containing all its elements (where $\lfloor a \rfloor$ denotes the largest F-number not greater than a and $\lceil b \rceil$ denotes the smallest F-number not smaller than b).

Real box, F-box and F-box approximation are extensions to several dimensions of the concepts of a real interval, F-interval and F-interval approximation. Real boxes and F-boxes will be denoted by tuples $<I_1,\ldots,I_n>$ of real intervals and F-intervals, respectively. A Real box is the Cartesian product of real intervals. An F-box is the Cartesian product of F-intervals, in particular, if all the F-intervals are canonical the F-box is *canonical*. The F-box approximation of a real box is the smallest F-box enclosing the real box.

We will call a *canonical solution* of a CCSP to any canonical F-box that cannot be proved inconsistent (wrt to the CCSP) either because it contains solutions or due to approximation errors in the evaluation of the constraint set.

2.2 Solving Continuous Constraint Satisfaction Problems

Solving a CCSP can be seen as a search process over the representable elements of the lattice of the variable domains. For a given CCSP a complete domain lattice L is defined by the elements obtained from the power set of the initial domains box, partially ordered by set inclusion ($\subseteq$) and closed under arbitrary intersections ($\cap$) and unions ($\cup$). A search procedure may start at the top of the domain lattice and navigate over the accessible elements of the lattice until eventually stopping, returning one of them. If it returns the bottom element (the empty set $\{\}$) then the CCSP has no solution. The navigation over the lattice elements usually alternates pruning with branching steps and ends whenever a stopping criterion is satisfied. To simplify the domains representation, most solving strategies, impose that the only elements considered in the pruning and branching steps, are representable by a single F-box.

Pruning consists of replacing an element of the lattice for a smaller element (wrt set inclusion) as a result of applying an appropriate filtering algorithm, which eliminates some value combinations inconsistent with the constraints. The filtering algorithm attempts to narrow the original F-Box into a smaller one (or prove its inconsistency) guaranteeing that no possible real solution is lost. If the filtering algorithm were complete, all inconsistent combinations of values would be deleted and the new top element would only contain the solution space. However, this is generally not the case, and several inconsistent combinations may still remain.

The branching step may be applied when the pruning step fails to further eliminate inconsistent combinations of values. The idea is to split a set of value combinations into smaller sets for applying latter the pruning step, hoping to get better filtering results on these reduced sets. The branching step usually consists on splitting the original F-box into two smaller F-boxes by splitting one of the variable domains.

Search over different branches may be done concurrently or by some backtracking mechanism until a stopping criterion is attained. This stopping criterion may be the achievement of the intended goal or the satisfaction of some specific properties imposed to avoid the complexity explosion of the search procedure. If the goal is the set of all solutions then the final result must be the union of the elements remaining at the end of the search process.

3 Interval Constraints

The Interval Constraints framework, firstly introduced by Cleary [3], combines Artificial Intelligence techniques with Interval Analysis methods for solving CCSPs.

Pruning of variable domains is based on constraint propagation techniques initially developed in Artificial Intelligence for finite domains. The main idea is to use partial information expressed by a constraint to eliminate some incompatible values from the domain of the variables within the scope of the constraint. Once the domain of a variable is reduced, this information is propagated to all constraints with that variable in their scopes, which must be checked again to possibly reduce further the domains of the other constrained variables. Propagation terminates when a fixed point is attained, that is, the variables domains cannot be further reduced by any constraint.

To reduce the domains of the variables appearing in a constraint, a filtering algorithm must enforce a local property relating their domains accordingly to the constraint. This property, imposed at the constraint level to each constraint, is called local consistency.

Interval Analysis, introduced by Moore in [11], enable the definition of sound local filtering algorithms. Moreover, efficient interval methods developed in Interval Analysis, (e.g. the interval Newton method [11, 7]), are used in interval constraints for producing efficient filtering algorithms.

3.1 Local Consistency

In CCSPs the enforcement of a local consistency wrt a constraint consists on narrowing an F-box (representing the domains of the variables within the constraint) into the largest F-box within the original one satisfying some local property.

The two main local consistencies used for solving CCSPs are hull-consistency [2] (or 2B-consistency [9]) and box-consistency [1, 15]. Both are approximations of arc-consistency [10], widely used in finite domains. Arc-consistency eliminates a value from a variable domain if it has no support, i.e. if no compatible values exist in the domain of the other variables sharing the same constraint. In continuous domains this kind of enumeration is not possible, so hull and box-consistency both assume that the domains of the variables are convex (they are represented by F-intervals), and aim at simply tightening their outer bounds. If the correct domains (wrt a constraint) are disjunctive both local consistencies admit inconsistent values within the outer limits of the variable domains.

The key idea behind hull-consistency is to guarantee arc-consistency only at the bounds of the variable domains. If a variable is instantiated with the value of one of its bounds then there must be a consistent instantiation of the other constraint variables. In practice, the result of enforcing hull-consistency on a box of domains is an F-box approximation of the real box that would be obtained by enforcing the above local property. The existing hull-consistency algorithms decompose the original constraint system into a set of primitive constraints (with the addition of extra variables), for which this property can be enforced by interval arithmetic operations.

The major drawback of this decomposition approach is the worsening of the locality problem. The existence of intervals satisfying a local property on each constraint does not imply the existence of value combinations satisfying simultaneously all of them. When a complex constraint is subdivided into primitive constraints this will only worsen this problem due to the addition of new variables and the consequent loss of dependency between values of related variables.

Consider constraint $x_1 \times (x_2 - x_1) = 0$ and F-boxes $A = <[0..2],[1..2]>$ and $B = <[-1..3],[1..2]>$. Clearly, any solution of the above constraint must be a point within the line $x_1 = 0$ or within the line $x_1 = x_2$. Therefore, box A, which is not arc-consistent (since $x_1 = 0.5$ has no support in $x_2 \in [1..2]$), is hull-consistent because each bound of each variable has a support in the other variable (e.g. $x_1 = 2$ has the support $x_2 = 2$ defining a point within the line $x_1 = x_2$). Moreover, A is the largest hull-consistent F-box within B and so it should be obtained by enforcing this property on B. However, the existing algorithms do not enforce hull-consistency directly in the above constraint, but on the decomposed set of primitives $x_1 \times x_3 = 0$ and $x_2 - x_1 = x_3$, introducing the new variable x_3. Because $B' = <[-1..3],[1..2],[-2..3]>$ is hull-consistent wrt the set of primitives, box B would not be narrowed by these algorithms.

Box-consistency guarantees the consistency of each bound of the domain of each variable with the F-intervals of the others. After substituting all but one variable by their interval domains, the problem of binding the remaining variable is tackled by numerical methods, in particular a combination of interval Newton iterates and bisection [1, 15]. The main advantage of this approach is that each constraint can be manipulated as a whole, not requiring the decomposition into primitive constraints, thus preventing the amplification of the locality problem. In the previous example the box B=<[-1..3],[1..2]> is not box-consistent wrt $x_1 \times (x_2 - x_1) = 0$ since, for example, the upper bound of x_1 is not consistent with x_2=[1..2] ($0 \notin 3\times([1..2]-3)=[-6..-3]$). Moreover, the enforcement of box-consistent on B wrt the original constraint would result in box A, which is the tightest box enclosure of its solution space within B.

The domain pruning obtained by box-consistency is often insufficient. The reason is that if there are n uncertain variables, it is still necessary to handle functions with n-1 interval values. Depending on the constraint, the uncertainty of the n-1 intervals may allow a wide range of possible values for these functions, preventing the finding of inconsistency required for domain reduction.

3.2 Higher Order Consistencies

Better pruning of the variable domains may be achieved if, complementary to a local property, some (global) properties are also enforced on the overall constraint set.

3B-consistency [9] and Bound-consistency [13] are higher order generalizations of hull and box-consistency respectively. In both, the property enforced on the overall constraint set is the following: if the domain of one variable is reduced to one of its bounds then this simplified CCSP must be local consistent (hull or box-consistent).

The algorithms to enforce these stronger consistencies are higher order propagation algorithms where constraint propagation is just a procedure that may be interleaved with search techniques. The price to pay for a stronger consistency requirement is the growth in the computational cost of the enforcing algorithm. The adequacy of a consistency criterion for a particular CCSP must be evaluated taking into account the trade-off between the pruning it achieves and its execution time. Moreover one must be aware that the filtering process is performed within a larger procedure for solving the CCSP and may be globally advantageous to obtain faster, if less accurate, results.

3.3 Octree Approach

An alternative approach to the interval constraints framework for solving CCSPs was proposed in [8]. The key idea is to compute the complete solution space of a CCSP through the propagation of total constraints represented by a set of partitions of the feasible space called 2^k-trees. A total constraint is a relation defining consistent value combinations among a set of variables, this relation summarizes all the constraints in the CCSP between the variables. Any CCSP can be represented by a set composed of ternary and binary total constraints. A 2^k-tree is a hierarchical decomposition of the solution space defined in the total constraint. The partition is achieved by recursively halving simultaneously each variable domain within a region that is not clearly within the feasible region. A maximum number of successive partitions is defined accordingly to a required precision. This precision, may be, in certain types of problems, increased to obtain better approximations of the solution space.

This partitioning approach is specially suited for dealing with CCSPs where all the relations determine convex regions. When regions are not convex, they must be decomposed into convex sub-regions, which implies the a priori definition of the required precision and can cause an explosion of disjoint solution regions.

4 Global Hull-Consistency

A stronger consistency, called global hull-consistency, was proposed in [4, 6] and applied to solve constraint problems with parametric ordinary differential equations. The key idea is to generalize local hull-consistency criterion to a higher level where the set of all constraints is seen as a single global constraint. Hence, it must guarantee arc-consistency at the bounds of the variable domains for this single global constraint. If a variable is instantiated with the value of one of its bounds then there must be a consistent instantiation of the other variables and this complete instantiation is a solution of the CCSP.

Again, due to the limitations of real value representation, the result of enforcing global hull-consistency on a box of domains must be an F-box enclosing the real box that would be obtained by enforcing the property. Because within a canonical solution there might be a real solution of the CCSP, the best thing that can be done is to guarantee that for each bound of each variable there is a canonical F-box instantiation which is a canonical solution. Hence, any strategy to enforce global hull-consistency must be able to localise the canonical solutions within a box of domains that are extreme with respect to each bound of each variable domain.

4.1 Enforcing Global Hull-Consistency with Existing Techniques

In the following we will consider three different approaches that can be easily implemented by existing constraint systems without significant modifications of their propagation mechanisms. We assume that there is a solving mechanism (alternating pruning and branching steps) implementing a backtracking search for obtaining canonical solutions. The pruning is achieved by enforcing a local consistency (hull or box-consistency).

The first approach (SOLVE$_1$) is a branch and bound algorithm where the extreme canonical solutions with respect to each bound of each variable domain are searched separately. To search for the extreme canonical solution with respect to the lower (upper) bound of the variable x_i the following algorithm is used:

(i) the solving mechanism is used to obtain a canonical solution of the CCSP;
(ii) if no canonical solution is found then the CCSP has no solutions;
(iii) if a new canonical solution $<I_1,\ldots,I_{i-1},[a..b],I_{i+1},\ldots,I_n>$ is found, the search space is pruned by imposing $x_i<a$ ($x_i>b$) and the solving proceeds by backtracking;
(iv) when the solving mechanism proves that there are no more canonical solutions, the last one found is guaranteed to be the extreme canonical solution with respect to the lower (upper) bound of the variable x_i

The second approach (SOLVE$_2$), suggested by Frédéric Benhamou (personal communication), controls the branching strategy of the solving mechanism to direct the search towards the extreme canonical solutions. Again the search is separately executed for each bound of each variable domain. To search for the extreme canonical solution wrt the lower (upper) bound of the variable x_i the strategy is:

(i) if the branching step is applied to a box $<I_1,\ldots,I_i,\ldots,I_n>$ where I_i is not canonical then obtain two F-boxes by splitting I_i around its mid value.

(ii) always search first the branch with smaller (larger) x_i values (the other branch is only searched by backtracking);
(iii) if no canonical solution is found then the CCSP has no solutions;
(iv) if a canonical solution is found then it is guaranteed to be the extreme canonical solution wrt to the lower (upper) bound of the variable x_i;

The third approach ($SOLVE_3$) is a branch and bound algorithm where the extreme canonical solutions with respect to each bound of each variable domain are searched simultaneously within a round robin scheme. The basic idea is to keep track of an outer box that must include all possible canonical solutions and an inner box that is the smallest box enclosing all the currently found canonical solutions. Initially the solving mechanism is applied to the original domain box to obtain a first canonical solution. If no canonical solution is found then the CCSP has no solutions. Otherwise the inner box is initialised to the obtained canonical solution and the outer box is initialised to the original domain box. The algorithm proceeds by alternating in a round robin fashion the search for the extreme canonical solution wrt the lower (upper) bound of each variable x_i:
(i) the solving mechanism is applied to a box obtained from the outer box by substituting the upper (lower) bound of its i^{th} domain by the lower (upper) bound of the inner box i^{th} domain;
(ii) if no canonical solution is found then the extreme canonical solution was already found and the lower (upper) bound of the outer box i^{th} domain is updated to the respective value in the inner box;
(iii) otherwise, the inner box is updated to include the new canonical solution;

The algorithm stops when the inner box equals the outer box guaranteeing that this is the largest global hull-consistent box within the original domains.

Enforcing global hull-consistency is a complex problem and this will be reflected in any algorithm to solve it. However, we argue that better alternatives may be devised to the above strategies. One explanation for the bad behaviour is that all these strategies rely on a solving mechanism developed for finding individual canonical solutions. This forces the strategies to repeatedly restart the search without completely profiting from the previous pruning effort.

4.2 The Tree Structured Algorithm

We propose an alternative algorithm for enforcing global hull-consistency based on a tree structured representation of the complete search space. During the whole process, the following data structures are maintained:
(i) a binary tree, where each node is an F-box representing a sub-region of the search space that may contain solutions of the CCSP. Each parent box is the smallest box enclosing its two children. The current search space is the union of all the leaves of the binary tree. The root is the smallest F-box enclosing the current search space;
(ii) an inner box, which is the smallest box enclosing all the canonical solutions found;

(iii) an ordered list associated with each bound of each variable. Each element of such list is a pair (*F-Box*, *Action*) where *Action* is a state representing the next action (prune, search or split) to perform on the *F-Box*. The list associated with the lower (upper) bound of variable x_i keeps track of the leaves of the binary tree in ascending (descending) order of their lower (upper) bounds for the x_i domain;

The algorithm alternates prune, search and split actions performed over specific sub-regions (*F*-boxes) of the current search space. The pruning of an *F*-box is achieved by enforcing a local consistency (hull or box-consistency). The search action is performed with the goal of finding a canonical solution within an *F*-box and may be implemented as a simple check of an initial guess or as a more complete local search procedure (see next section). The split of an *F*-box is done by splitting one of its variable domains (the one with the largest width) at the mid point.

The algorithm proceeds by considering, in round robin, the first element (*FB*,*A*) of each ordered list associated with the lower (upper) bound of each variable x_i:

(i) if *FB* and the inner box have the same lower (upper) bound of the i^{th} domain then do nothing (the respective extreme canonical solution was already found);

(ii) otherwise the current *F*-box to investigate is the sub-region of *FB* with x_i values smaller (larger) than the respective lower (upper) bound in the inner box (if no canonical solution was yet found then the current *F*-box is *FB*);

(iii) action *A* is performed over the current *F*-box and its consequences must be propagated over the data structures maintained by the algorithm:

  (a) pruning: The search space discarded in the pruning must be removed from the binary tree and in particular from *FB*. This may imply to narrow *FB*, to eliminate *FB* or split *FB* into the new narrowed sub-region and the non-investigated sub-region. Changes on the leaves of the binary tree imply that the associated elements in the ordered lists may need to be reordered, eliminated or inserted. In case of a new element the information about the next action is initialised to pruning. Otherwise, after pruning, the next action is updated to searching.

  (b) search: If a canonical solution was not found then *A* is updated to splitting. Otherwise, the inner box is updated to include this new canonical solution and the elements of the ordered lists with sub-regions including it must reset their actions to pruning.

  (c) split: The binary tree must be updated with the new leaves which are descendants of *FB*. The creation of new leaves implies the insertion of new elements in the ordered lists with actions initialised to pruning.

The algorithm stops when the inner box equals the root box guaranteeing that this is the largest global hull-consistent box within the original domains.

The main advantage of the proposed algorithm is the dynamic representation of the search space allowing the focussing on regions that are relevant for the enforcement of global hull-consistency without losing information obtained in the pruning process.

Another advantage wrt the previous approaches regards the better results obtained by an any-time use of the algorithm. In the other strategies, the box enclosing all possible solutions could only be narrowed by finding a canonical solution and proving it to be a extreme wrt a bound of a variable. This complete proof can be extremely time consuming, despite some outer regions of the search space might be easily proved inconsistent. The tree-structured algorithm firstly concentrates the pruning on the outer regions of the search space delaying the analysis of the internal more difficult regions. Any pruning achieved is propagated immediately along the binary tree, maintaining, at the root, an F-box enclosing the current search space, which may be returned at any time in the process. Moreover, this box may be compared with the current inner box to bind the narrowing expectations.

Finally, the tree-structured representation may be used for further analysis, since it is a more detailed representation of the solution space than just an enclosing box. It could help to find interactively important possible sub-regions of the search space. Another possibility is to enforce a stronger consistency criterion based on the recursive enforcement of global hull-consistency on each leaf of the tree.

5 Local Search

An important characteristic of the tree-structured algorithm is to explore only relevant sub-regions of the search space that are outside the inner box. Consequently the discovery of a new canonical solution reduces the relevant search space as a result of the enlargement of the inner box. Applying local search techniques for finding canonical solutions within a box may significantly enhance the algorithm efficiency.

The key idea of local search techniques is to navigate on points of the search space by inspecting some local properties of the current point to choose a nearby point to jump to. We developed the following local search algorithm for finding canonical solutions of a CCSP within a search box *FB*:

(i) initially a starting point is chosen to be the current point. If the goal is to find the lower (upper) bound of variable x_i, the point is the mid point of *FB* except that the i^{th} domain is the smallest (largest) x_i value within the search box;
(ii) if the current point is within a canonical solution then the algorithm stops;
(iii) otherwise, a multidimensional vector is obtained based on the Newton's method for multidimensional root finding (see next subsection);
(iv) a minimization process (described in subsection 5.2) obtains a new point inside the search box and within the line segment defined by the current point and the point obtained by applying the multidimensional vector to the current point.
(v) if the distance between the new point and the current point does not exceed a convergence threshold then the algorithm stops without finding any solution;
(vi) otherwise the current point is updated to the new point and the algorithm proceeds in step (ii);

The algorithm is guaranteed to stop since step (iv) ensures the minimization of a function for which any canonical solution is a zero and the convergence to a local minimum is detected in step (v).

5.1 Obtaining a Multidimensional Vector

The ultimate goal of obtaining a multidimensional vector is to find a solution by applying it to the current point. The idea is to reduce the problem of finding a solution into the problem of finding a root of a multidimensional vector function $\boldsymbol{F}$. The vector function must be associated with the set of constraints in a way that a zero of each element F_i must satisfy the constraint C_i. For example, if the constraints are $x_1^2+x_2^2=1$ and $x_1\times(x_2-x_1)=0$ then $F_1(x_1,x_2)=x_1^2+x_2^2-1$ and $F_2(x_1,x_2)=x_1\times(x_2-x_1)$.

The Newton's iterative method for multidimensional root finding [12] is known to rapidly converge to a root from a sufficiently good initial guess. The idea is to compute a multidimensional vector $\boldsymbol{\delta x}$ (the Newton vector) which applied to current point $\boldsymbol{x}$ reaches a root of the multidimensional function $\boldsymbol{F}(\boldsymbol{x}+\boldsymbol{\delta x})=\boldsymbol{0}$. Expanding the function in Taylor series and neglecting the higher order terms: $\boldsymbol{F}(\boldsymbol{x}+\boldsymbol{\delta x})=\boldsymbol{F}(\boldsymbol{x})+\boldsymbol{J}\cdot\boldsymbol{\delta x}$ (where $\boldsymbol{J}$ is the Jacobian matrix). Finally if $\boldsymbol{x}+\boldsymbol{\delta x}$ is a root, then $\boldsymbol{J}\cdot\boldsymbol{\delta x} = -\boldsymbol{F}(\boldsymbol{x})$.

Solving this equation in order to $\boldsymbol{\delta x}$ we obtain a estimation of the corrections to the current point $\boldsymbol{x}$ that move each function $F_i(\boldsymbol{x})$ closer to zero simultaneously. To solve the equation, it is convenient to use a numerical technique called Singular Value Decomposition (SVD) [14] since it can be used to solve any set of linear equations.

The equation may have zero, one or several solutions. If there are no solutions the SVD technique computes the value of $\boldsymbol{\delta x}$ that minimizes the distance between the two sides of the equation which is the best possible correction. If there are one or more solutions, the SVD technique computes the $\boldsymbol{\delta x}$ that is closer to zero (minimizes $|\boldsymbol{\delta x}|$) which is the smallest possible correction.

5.2 Obtaining a New Point

One problem of the Newton's iterative method is that it may fail convergence if the initial guess is not good enough. To correct this we followed a globally convergent strategy that guarantees some progress toward the solution at each iteration. The fact that the Newton vector $\boldsymbol{\delta x}$ defines a descent direction for $|\boldsymbol{F}|^2$ guarantees that it is always possible to obtain along that direction a point closer to a zero of $\boldsymbol{F}$. This new point must lie in the segment defined by $\boldsymbol{x}+\lambda\cdot\boldsymbol{\delta x}$ with $\lambda\in[0..1]$. The strategy to obtain the new point consists of trying different λ values, starting with the largest possible value without exceeding the search box limits, and backtracking to smaller values until a suitable point is reached. The idea is that if $\boldsymbol{x}$ is close enough to the solution then the Newton step has quadratic convergence. Otherwise a smaller step is taken still directed to a solution (or a mere local minimum), guaranteeing convergence.

In alternative to the Newton approach, other minimization methods (as the Conjugate Gradient Method [14]) could have been directly applied to the scalar function $|\boldsymbol{F}|^2$. However, the early collapsing of the various dimensions of $\boldsymbol{F}$ into a single one implies the lack of information about each individual constraint and makes these strategies more vulnerable to local minima.

6 An Under-Constrained CCSP

In this section we illustrate the concept of global hull-consistency with a simple example of an under-constrained CCSP. There are two variables x_1 and x_2 with real values ranging within [-5..5] and constrained by: $x_1^2+x_2^2\leq 2^2$ and $(x_1-1)^2+(x_2-1)^2\geq 2.5^2$.

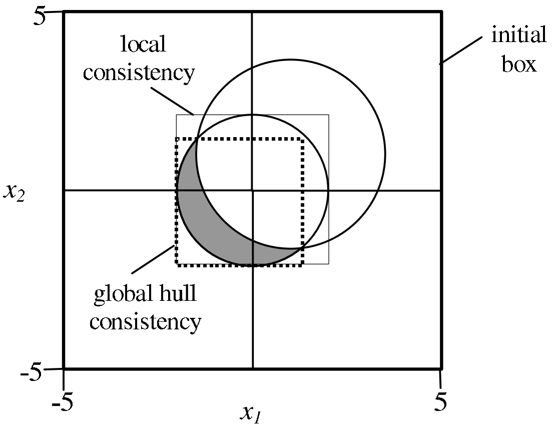


Fig. 1. Local consistency and global hull-consistency on a simple CCSP.

Fig. 1 sketches the problem. The thick solid square is the initial domain box. The two circumferences represent the two constraints. The grey area represents the complete set of solutions. The thin solid square is the box obtained by enforcing a local consistency, either box-consistency or hull-consistency (on the decomposed problem). The dashed square is the box obtained by enforcing global hull-consistency.

The figure shows that the local consistency criterion cannot prune the search space inside the smaller circumference – the pruning is the same as it would be without the constraint associated with the larger circumference. Depending on the decision problem to solve, this may be irrelevant or, on the contrary, it may justify a more time consuming enforcement of a stronger consistency, such as global hull-consistency.

Better pruning is obtained by applying the octree approach (see section 3.3), whose goal is much more ambitious, aiming at providing a compact description of the complete set of solutions. To achieve this goal the entire solution space is extensively partitioned. Fig. 2 compares the partitions obtained by this technique (with at most 6 subdivisions) with the partitions obtained by the tree structured algorithm with local search for global hull-consistency (with 10^{-8} precision limit –see next section).

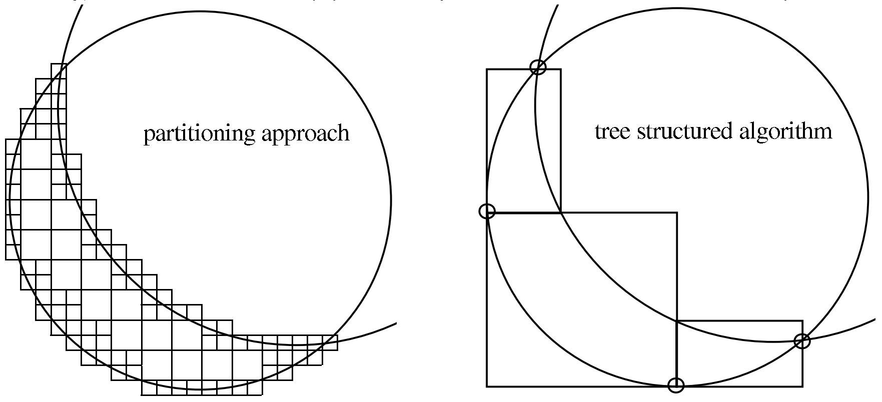


Fig. 2. Pruning by an extensive partitioning approach and the tree structured algorithm.

Only 3 boxes (leaves of the binary tree, the complete tree has a total of 5 boxes) needed to be considered for enforcing global hull-consistency since the 4 extreme canonical solutions (represented in the figure as small circles) were immediately found. In the other approach 92 partitions had to be considered delimiting with a precision of 0.15625 (the size of smaller squares) the all boundary of the feasible region. Note that in terms of a single box enclosing the complete set of solutions this approach is much less precise than the global hull-consistency approach

The above discussion justifies our belief that global hull-consistency may be a good compromise between less accurate local consistencies and extensive, but too demanding, partitioning approaches.

6.1 Preliminary Results

We now compare the performance of the tree-structured algorithm for enforcing global hull-consistency with the alternative algorithms discussed in section 4.1 ($SOLVE_1$, $SOLVE_2$ and $SOLVE_3$). Two versions of the algorithm are analysed: version TSA_0 has no local search (only an initial guess is tried) whereas version TSA_{ls} implements local search as described in section 5. All algorithms use the same local consistency criterion (box-consistency) for pruning each box domain. An extension of the previous CCSP example to three dimensions was considered since the version with two dimensions was easily solved by several of the above approaches. Several different precision limits (ε) were also considered for the smallest acceptable sizes of the domain boxes. Table I shows the results.

Table I – Execution times for different precision limits.

	$SOLVE_1$	**$SOLVE_2$**	**$SOLVE_3$**	**TSA_0**	**TSA_{ls}**
$\varepsilon=10^{-3}$	1min 3.360s	4.010s	7.720s	0.660s	0.080s
$\varepsilon=10^{-6}$	> 15min	2min 6.720s	4min 19.220s	17.600s	1.700s
$\varepsilon=10^{-8}$	> 15min	10min 37.560s	> 15min	8min 23.800s	11.930s

The TSA_{ls} algorithm clearly outperformed all the other approaches. Moreover, the version without local search (TSA_0) was distinctly inferior for solving this problem, despite exibiting better efficiency than the best of the $SOLVE_i$ algorithms. The enhancement obtained by local search is a direct consequence of the reduction on the domains partitioning. Table II compares the number of leaves in the binary tree structure at the end of both algorithms (TSA_0 and TSA_{ls}).

Table II – Number of leaves in the binary tree structure for different precision limits.

	$\varepsilon=10^{-3}$	$\varepsilon=10^{-6}$	$\varepsilon=10^{-8}$
TSA_0	309	9093	91134
TSA_{ls}	23	77	316

7 Conclusions

This paper addressed constraint solving over continuous domains in the context of decision making, and discusses the trade-off between precision in the definition of the solution space and the computational efforts required. We showed that in this context, our proposed approach to enforce a global hull consistency may be an appropriate choice, achieving acceptable precision with relatively low computational cost.

Such effort depends on the algorithms used to enforce such consistency. Among the set of algorithms we developed, the one that integrates local search with constraint propagation, within a tree-structured representation of the domain, has clearly shown the best performances, which makes a case to further research the integration of constraint propagation and local search methods within interval constraint solving.

Although the results were obtained in a particular, and simple, example, we believe that this approach can scale up to more complex and realistic problems. This is clearly a direction for further research that we intend to proceed, namely to extend our previous work on continuous constraints with differential equations [5].

Acknowledgements. This work was partly supported by project PROTEIN – POSI/1999/SRI/33794 granted by the Portuguese Foundation for Science and Technology. Special thanks to F. Benhamou, for the valuable discussions.

References

1. Benhamou, F., McAllester, D., Van Hentenryck, P.: CLP(Intervals) revisited. *Logic Programming Symposium*, MIT Press (1994) 124-131.
2. Benhamou, F., Older, W.J.: Applying Interval Arithmetic to Real, Integer and Boolean Constraints. *Journal of Logic Programming* (1997) 32(1): 1-24.
3. Cleary, J.G.: Logical Arithmetic. *Future Computing Systems* (1987) 2(2):125-149.
4. Cruz, J., Barahona, P.: An Interval Constraint Approach to Handle Parametric ODEs for Decision Support. *Principles Practice Constraint Programming*, Springer (1999) 478-479.
5. Cruz, J., Barahona, P., Benhamou, F.: Integrating Deep Biomedical Models into Medical DSSs: an Interval Constraint Approach. *AI in Medicine*, Springer (1999) 185-194.
6. Cruz, J., Barahona, P.: Handling Differential Equations with Constraints for Decision Support. *Frontiers of Combining Systems*, Springer (2000) 105-120.
7. Hansen, E.: Global Optimization Using Interval Analysis. Marcel Dekker (1992).
8. Sam-Haroud, D., Faltings, B.V.: Consistency Techniques for Continuous Constraints. *Constraints* (1996) 1(1,2):85-118.
9. Lhomme, O.: Consistency Techniques for Numeric CSPs. *IJCAI*, IEEE Pr. (1993) 232-238.
10. Montanari U.: Networks of Constraints: Fundamental Properties and Applications to Picture Processing. *Information Science* (1974) 7(2):95-132.
11. Moore, R.E.: Interval Analysis. Prentice-Hall (1966).
12. Ortega, J., Rheinboldt, W.: Iterative Solution of Nonlinear Equations in Several Variables. Academic Press (1970).
13. Puget, J-F., Van Hentenryck, P.: A Constraint Satisfaction Approach to a Circuit Design Problem. *Journal of Global Optimization*, MIT Press (1997).
14. Stoer, J., Burlisch R.: Introduction to Numerical Analysis. Springer Verlag (1980).
15. Van Hentenryck, P., McAllester, D., Kapur, D.: Solving Polynomial Systems Using a Branch and Prune Approach. *SIAM Journal of Numerical Analysis* (1997) 34(2).

Towards Provably Complete Stochastic Search Algorithms for Satisfiability

Inês Lynce, Luís Baptista, and João Marques-Silva

Department of Informatics,
Technical University of Lisbon,
IST/INESC/CEL, Lisbon, Portugal
{ines,lmtb,jpms}@sat.inesc.pt

Abstract. This paper proposes a stochastic, and complete, backtrack search algorithm for Propositional Satisfiability (SAT). In recent years, randomization has become pervasive in SAT algorithms. Incomplete algorithms for SAT, for example the ones based on local search, often resort to randomization. Complete algorithms also resort to randomization. These include, state-of-the-art backtrack search SAT algorithms that often randomize variable selection heuristics. Moreover, it is plain that the introduction of randomization in other components of backtrack search SAT algorithms can potentially yield new competitive search strategies. As a result, we propose a stochastic backtrack search algorithm for SAT, that randomizes both the variable selection and the backtrack steps of the algorithm. In addition, we describe and compare different organizations of stochastic backtrack search. Finally, experimental results provide empirical evidence that the new search algorithm for SAT results in a very competitive approach for solving hard real-world instances.

1 Introduction

Propositional Satisfiability is a well-known NP-complete problem, with extensive applications in Artificial Intelligence, Electronic Design Automation, and many other fields of Computer Science and Engineering.

In recent years, several competitive solution strategies for SAT have been proposed and thoroughly investigated [10,9,11]. Advanced techniques applied to backtrack search algorithms for SAT have achieved remarkable improvements [3,7,9,11,12,15], having been shown to be crucial for solving large instances of SAT derived from real-world applications. Current state-of-the-art SAT solvers incorporate advanced pruning techniques as well as new strategies on how to organize the search. Effective search pruning techniques include, among others, clause recording and non-chronological backtracking [3,9,11], whereas recent effective strategies include search restart strategies [7]. Moreover, the work of S. Prestwich [12] (inspired by the previous work of others [6,13]) has motivated the utilization of randomly picked backtrack points in incomplete SAT algorithms. More recently, a stochastic systematic search algorithm has been proposed [8].

The remainder of this paper is organized as follows. Section 2 briefly surveys SAT algorithms and the utilization of randomization in SAT. Afterwards, Section 3 introduces a stochastic backtrack search SAT algorithm and the next section details randomized backtracking. Preliminary experimental results are presented and analyzed in Section 5. Finally, we conclude and suggest future research work in Section 6.

2 SAT Algorithms

Over the years a large number of algorithms have been proposed for SAT, from the original Davis-Putnam procedure [5], to recent backtrack search algorithms [3,9,11,15], among many others.

SAT algorithms can be characterized as being either *complete* or *incomplete*. Complete algorithms can establish unsatisfiability if given enough CPU time; incomplete algorithms cannot. In a search context complete and incomplete algorithms are often referred to as *systematic*, whereas incomplete algorithms are referred to as *non-systematic*.

The vast majority of backtrack search SAT algorithms build upon the original backtrack search algorithm of Davis, Logemann and Loveland [4]. A generic organization of backtrack search for SAT considers three main engines:

– The decision engine (`Decide`) which selects an elective variable assignment each time it is called.
– The deduction engine (`Deduce`) which applies Boolean Constraint Propagation, given the current variable assignments and the most recent decision assignment, for satisfying the CNF formula.
– The diagnosis engine (`Diagnose`) which identifies the causes of a given conflicting partial assignment.

Recent state-of-the-art backtrack search SAT solvers [3,9,11,15] utilize sophisticated variable selection heuristics, implement fast Boolean Constraint Propagation procedures, and incorporate techniques for diagnosing conflicting conditions, thus being able to backtrack non-chronologically and record clauses that explain and prevent identified conflicting conditions. Clauses that are recorded due to diagnosing conflicting conditions are referred to as *conflict-induced clauses* (or simply *conflict clauses*).

3 Stochastic Systematic Search

In this section we describe how randomization can be used within backtrack search algorithms to yield a *stochastic systematic search* SAT algorithm.

As previously explained in Section 2, a backtrack search algorithm can be organized according to three main engines: the decision engine, the deduction engine and the diagnosis engine. Given this organization, we define a backtrack search (and so systematic) SAT algorithm to be *stochastic* provided all three engines are subject to randomization:

1. Randomization can be (and has actually been [2,3,11]) applied to the decision engine by randomizing the variable selection heuristic.
2. Randomization can be applied to the deduction engine by randomly picking the order in which implied variable assignments are handled during Boolean Constraint Propagation.
3. The diagnosis engine can be randomized by randomly selecting the point to backtrack to.

For the deduction engine, randomization only affects the order in which assignments are implied, and hence can only affect which conflicting clause is identified first, and so it is not clear whether randomization of the deduction engine can play a significant role. As a result, we chose to randomize the two other engines of the backtrack search SAT algorithm.

Since the randomization of the decision engine is simply obtained by randomizing the variable selection heuristic [2,3,11], in the next section we focus on the randomization of the diagnosis engine.

4 Randomized Backtracking

State-of-the-art SAT solvers currently utilize different forms of non-chronological backtracking, for which each identified conflict is analyzed, its causes identified, and a new clause created and added to the CNF formula. Created clauses are then used to compute the backtrack point as the *most recent* decision assignment from all the decision assignments represented in the recorded clause.

The diagnosis engine of a non-chronological backtrack search algorithm can be randomized by randomly selecting the point to backtrack to. The conflict clause is then used for *randomly* deciding which decision assignment is to be toggled. This form of backtracking is referred to as *random backtracking*.

In SAT solvers implementing non-chronological backtracking and clause recording, even with opportunistic clause deletion, the algorithms are guaranteed to be complete, because there is always an implicit explanation for why a solution cannot be found in the portion of the search space already searched. However, in order to relax this backtracking condition and still ensure completeness, randomized backtracking requires that *all* recorded clauses must be kept in the CNF formula.

Moreover, there exists some freedom on *how* the backtrack step to the target decision assignment variable is performed and on *when* it is applied. For example, one can decide not to apply randomized backtracking after every conflict but instead only once after every K conflicts.

4.1 Completeness Issues

With randomized backtracking, clause deletion may cause already visited portions of the search space to be visited again. A simple solution to this problem is to prevent deletion of recorded clauses, i.e. no recorded conflict clauses are ever deleted. If no conflict clauses are deleted, then conflicts cannot be repeated, and the backtrack search algorithm is necessarily complete. The main drawback of keeping all recorded clauses is that the growth of the CNF formula is linear in the number of explored nodes, and so exponential in the number of variables. However, as will be described in Section 4.2, there are effective techniques to tackle the potential exponential growth of the CNF formula. Moreover, experimental data from Section 5 clearly indicate that the growth of the CNF formula is not exponential in practice.

It is important to observe that there are other approaches to ensure completeness that do not necessarily keep all recorded conflict clauses:

1. One solution is to increase the value of K each time a randomized backtrack step is taken.
2. Another solution is to increase the relevance-based learning [3] threshold each time a randomized backtrack step is taken (i.e. after K conflicts).
3. One final solution is to increase the size of recorded conflict clauses each time a randomized backtrack step is taken.

Observe that all of these alternative approaches guarantee that the search algorithm is eventually provided with enough space and/or time to either identify a solution or prove unsatisfiability. However, all strategies exhibit a key drawback: *paths in the search tree can be visited more than once.* Moreover, even when recording of conflict clauses is used, as in [9,11], clauses can eventually be deleted and so search paths may be re-visited.

We should note that, as stated earlier in this section, if *all* recorded clauses are kept, then no conflict can be repeated during the search, and so no search paths can be repeated. Hence, as long as the search algorithm keeps *all* recorded conflict clauses, no search paths are ever repeated.

4.2 Implementation Issues

After (randomly) selecting a backtrack point, the actual backtrack step can be organized in two different ways:

- One can *non-destructively* toggle the target decision assignment, meaning that all other decision assignments are unaffected.
- One can *destructively* toggle the target decision assignment, meaning that all of the more recent decision assignments are erased.

The two randomized backtracking approaches differ significantly. Destructive randomized backtracking is more drastic and attempts to rapidly cause the search to explore other portions of the search space. Non-destructive randomized backtracking has characteristics of local search, in which the current (partial) assignment is only locally modified.

Another significant implementation issue is memory growth. Despite the growth of the number of clauses being linear in the number of searched nodes, for some problem instances a large number of backtracks will be required. However, there are effective techniques to tackle the potential exponential growth of the CNF formula. Next we describe two of these techniques:

1. The first technique for tackling CNF formula growth is to opportunistically apply subsumption to recorded conflict clauses. This technique is guaranteed to effectively reduce the number of clauses that are kept in between randomized backtracks.
2. Alternatively, a second technique consists of *just* keeping recorded conflict clauses that explain why each sub-tree, searched in between randomized backtracks, does not contain a solution. This process is referred to as identifying the *tree signature* [1] of the searched sub-tree.

Regarding the utilization of tree signatures, observe that it is always possible to characterize a tree signature for a given sub-tree T_S that has just been searched by the algorithm. Each time, after a conflict is identified and a randomized backtrack step is to be taken, the algorithm defines a path in the search tree. Clearly, the explanation for the current conflict, as well as the explanations for all of the conflicts in the search path, provide a *sufficient* explanation of why sub-tree T_S, that has just been searched, does not contain a solution to the problem instance.

4.3 Randomized Backtracking and Search Restart Strategies

It is interesting to observe that randomized backtracking strategies can be interpreted as a generalization of search restart strategies. The latter always start the search process from the root of the search tree, whereas the former randomly select the point in the search tree from which the search is to be restarted (assuming destructive backtracking is used). Moreover, observe that both approaches impose the same requirements in terms of completeness, and that the alternative techniques for completeness described in Section 4.1 for random backtracking also apply to search restart strategies.

It is also interesting to observe that the two strategies can be used together. In this case, each strategy S_{rb} (for randomized backtracking) or S_{rst} (for restarts) is applied after every K_{rb} or K_{rst} conflicts, respectively. In general we assume $K_{rb} < K_{rst}$, since S_{rst} causes the search to explore new portions of the search space that differ more drastically from those explored by S_{rb}.

5 Experimental Results

This section presents and analyzes experimental results that evaluate the effectiveness of the techniques proposed in this paper in solving hard real-world problem instances. Recent examples of such instances are the superscalar processor verification problem instances developed by M. Velev and R. Bryant [14]. We consider four sets of instances: *sss1.0a* with 9 satisfiable instances, *sss1.0* with 40 selected satisfiable instances, *sss2.0* with 100 satisfiable instances, and *sss-sat-1.0* with 100 satisfiable instances. For all the experimental results presented in this section a PIII @ 866MHz Linux machine with 512 MByte of RAM was used. The CPU time limit for each instance was set to 200 seconds, except for the *sss-sat-1.0* instances for which it was set to 1000 seconds. Since randomization was used, the number of runs was set to 10 (due to the large number of problem instances being solved). Moreover, the results shown correspond to the median values for all the runs.

In order to analyze the different techniques, a new SAT solver — Quest0.5 — has been implemented. Quest0.5 is built on top of the GRASP SAT solver [9], but incorporates restarts as well as *random backtracking*. Random backtracking is applied non-destructively after every K *backtracks* [1]. Furthermore, in what concerns implementation issues (see section 4.2), the backtracking point is selected from the union of the recorded conflict clauses in the most recent K conflicts and the tree signature of each sub-tree is kept in between randomized backtracks.

Moreover, for the experimental results, a few configurations were selected:

– **Rst1000+inc100** indicates that restarts are applied after every 1000 backtracks (i.e. the initial cutoff value is 1000), and the increment to the cutoff value after each restart is 100 backtracks. (Observe that this increment is necessary to ensure completeness.)
– **Rst1000+ts** configuration also applies restarts after every 1000 backtracks and keeps the clauses that define the tree signature when the search is restarted. Moreover, the cutoff value used is 1000, being kept fixed, since completeness is guaranteed.
– **RB1** indicates that random backtracking is taken at every backtrack step;
– **RB10** applies random backtracking after every 10 backtracks;
– **Rst1000+RB1** means that random backtracking is taken at every backtrack and that restarts are applied after every 1000 backtracks. (The identification of the tree signature is used for both randomized backtracking and for search restarts.)
– **Rst1000+RB10** means that random backtracking is taken after every 10 backtracks and also that restarts are applied after every 1000 backtracks. (The identification of the tree signature is used for both randomized backtracking and for search restarts.)

The results for Quest0.5 on the SSS instances are shown in Table 1. In this table, *Time* denotes the CPU time, *Nodes* the number of decision nodes, and X the average number of aborted problem instances. As can be observed, the results for Quest0.5 reveal interesting trends:

– Random backtracking taken at every backtrack step allows significant reductions in the number of decision nodes.
– The elimination of repeated search paths in restarts, when based on identifying the tree signatures and when compared with the use of an increasing cutoff value, helps reducing the total number of nodes and CPU time.
– The best results are always obtained when random backtracking is used, independently of being or not used together with restarts.

[1] For Quest0.5 we chose to use the number of backtracks instead of the number of conflicts. original GRASP code is organized [9].

Table 1. Results for the SSS instances

Inst	sss1.0a			sss1.0			sss2.0			sss-sat-1.0		
Quest 0.5	Time	Nodes	X	Time	Nodes	X	Time	Nodes	X	Time	Nodes	X
Rst1000+inc100	208	59511	0	508	188798	0	1412	494049	0	50512	8963643	39
Rst1000+ts	161	52850	0	345	143735	0	1111	420717	0	47334	7692906	28
RB1	79	11623	0	231	29677	0	313	31718	0	10307	371277	1
RB10	204	43609	0	278	81882	0	464	118150	0	6807	971446	1
Rst1000+RB1	79	11623	0	221	28635	0	313	31718	0	10330	396551	2
Rst1000+RB10	**84**	**24538**	**0**	**147**	**56119**	**0**	**343**	**98515**	**0**	**7747**	**1141575**	**0**
GRASP	1603	257126	8	2242	562178	11	13298	3602026	65	83030	12587264	82

– **Rst1000+RB10** is the only configuration able to solve all the instances in the allowed CPU time for *all* runs.

The experimental results reveal additional interesting patterns. When compared with the results for GRASP, Quest 0.5 yields dramatic improvements. Furthermore, even though the utilization of restarts reduces the amount of search, it is also clear that more significant reductions can be achieved with randomized backtracking. In addition, the integrated utilization of search restarts and randomized backtracking allows obtaining the best results, thus motivating the utilization of multiple search strategies in backtrack search SAT algorithms.

6 Conclusions and Future Work

This paper proposes and analyzes the application of randomization in the different components of backtrack search SAT algorithms. A new, stochastic but complete, backtrack search algorithm for SAT is proposed.

In conclusion, the contributions of this paper can be summarized as follows:

1. A new backtrack search SAT algorithm is proposed, that randomizes the variable selection and the backtrack steps.
2. The proposed SAT algorithm is shown to be complete, and different approaches for ensuring completeness are described.
3. Randomized backtracking is shown to be a generalization of search restart strategies, and their joint utilization is proposed.
4. Experimental results clearly indicate that significant savings in search effort can be obtained for different organizations of the proposed algorithm.

In the near future, we expect to consider other variations of this new algorithm. We can envision establishing a generic framework for implementing backtracking strategies, allowing the implementation of different hybrids, all guaranteed to be complete and so capable of proving unsatisfiability.

References

1. L. Baptista, I. Lynce, and J. Marques-Silva. Complete search restart strategies for satisfiability. In *IJCAI Workshop on Stochastic Search Algorithms*, August 2001.
2. L. Baptista and J. P. Marques-Silva. Using randomization and learning to solve hard real-world instances of satisfiability. In *International Conference on Principles and Practice of Constraint Programming*, pages 489–494, September 2000.
3. R. Bayardo Jr. and R. Schrag. Using CSP look-back techniques to solve real-world SAT instances. In *Proceedings of the National Conference on Artificial Intelligence*, pages 203–208, 1997.
4. M. Davis, G. Logemann, and D. Loveland. A machine program for theorem-proving. *Communications of the Association for Computing Machinery*, 5:394–397, July 1962.
5. M. Davis and H. Putnam. A computing procedure for quantification theory. *Journal of the Association for Computing Machinery*, 7:201–215, 1960.
6. M. Ginsberg and D. McAllester. GSAT and dynamic backtracking. In *Proceedings of the International Conference on Principles of Knowledge and Reasoning*, pages 226–237, 1994.
7. C. P. Gomes, B. Selman, and H. Kautz. Boosting combinatorial search through randomization. In *Proceedings of the National Conference on Artificial Intelligence*, July 1998.
8. I. Lynce, L. Baptista, and J. Marques-Silva. Stochastic systematic search algorithms for satisfiability. In *LICS Workshop on Theory and Applications of Satisfiability Testing*, June 2001.
9. J. P. Marques-Silva and K. A. Sakallah. GRASP-A search algorithm for propositional satisfiability. *IEEE Transactions on Computers*, 48(5):506–521, May 1999.
10. D. McAllester, B. Selman, and H. Kautz. Evidence of invariants in local search. In *Proceedings of the National Conference on Artificial Intelligence*, pages 321–326, August 1997.
11. M. Moskewicz, C. Madigan, Y. Zhao, L. Zhang, and S. Malik. Engineering an efficient SAT solver. In *Proceedings of the Design Automation Conference*, 2001.
12. S. Prestwich. A hybrid search architecture applied to hard random 3-sat and low-autocorrelation binary sequences. In *Proceedings of the International Conference on Principles and Practice of Constraint Programming*, pages 337–352, September 2000.
13. E. T. Richards and B. Richards. Restart-repair and learning: An empirical study of single solution 3-sat problems. In *CP Workshop on the Theory and Practice of Dynamic Constraint Satisfaction*, 1997.
14. M. N. Velev and R. E. Bryant. Superscalar processor verification using efficient reductions from the logic of equality with uninterpreted functions to propositional logic. In *Proceedings of Correct Hardware Design and Verification Methods*, LNCS 1703, pages 37–53, September 1999.
15. H. Zhang. SATO: An efficient propositional prover. In *Proceedings of the International Conference on Automated Deduction*, pages 272–275, July 1997.

A Combined Constraint-Based Search Method for Single-Track Railway Scheduling Problem

Elias Oliveira[1]* and Barbara M. Smith[2]

[1] School of Computing, University of Leeds, Leeds LS2 9JT, U.K.
elias@comp.leeds.ac.uk
[2] School of Computing & Mathematics, University of Huddersfield, Huddersfield HD1 3DH, U.K.
b.m.smith@hud.ac.uk

Abstract. In this paper we present a three-phase structure algorithm devised within a constraint programming paradigm to solve real-life single-track railway scheduling instances of problems. The combination of a hill-climbing, easing the process of finding iteratively improved solutions, and a branch-and-bound strategies allows us to solve 21 real-life problems in a reasonable time, 19 of them to optimality.
In addition, this paper discusses a group of practical constraints, incorporated into the software, that arise in real-life problems to which little attention has hitherto been paid. Results comparing the gain on using this approach on large instances problems are also presented.

1 Introduction

The railway services have increasingly become more competitive and so they demand more new features in the planning process. Hence, a tool to help the planner to meet changes in passenger demand is desirable.

This work is concerned with the single-track railway scheduling problem where trains are only allowed to pass one another at stations, sidings and double-track sections. In this paper both stations, sidings and double-track sections will be named *passing points* and, therefore, will be considered as similar entities for the model proposed in Section 2. The single-track sections of the line, between passing points, are divided into track segments, and at most one train can be on any track segment at any time.

Unlike a track segment, a passing point has a specified limit > 1 on the number of trains it can hold at any one time. A conflict can occur when two or more train services are planned to use the same track segment at the same time. Likewise, at a passing point a conflict occurs if its capacity is exceeded.

In order to resolve a conflict, one of the conflicting train must be delayed at that track section or passing point.

We present in this paper a model which maps this problem into a special case of job-shop scheduling problem. The objective function is to minimize the total tardiness of the trips [8].

* The author is a PhD student funded by CAPES under Grant BEX–1162/96-9.

A hill-climbing procedure is devised which uses the critical path structure of a solution of the problem and iteratively improve the solution by swapping pairs of *critical tasks*. This local search procedure is helped by a branch-and-bound algorithm when is trapped in a local minimum.

The single-line track railway scheduling problem has been tackled by modelling it as a Mathematical Integer Programming (MIP) [5, 3] problem. Considering the NP-completeness of this problem [2], some Meta-heuristic approaches have also been proposed [2].

The aim of previous authors were basically to resolve the conflicts between trains arising from a given desirable timetable [6]. However, there are other constraints which train-operating companies would like to see also incorporated into a tool. This would provide them with the possibility of offering better planning services. Therefore, this tool must be capable of both resolving conflicts and satisfying the additional constraints.

The aim of this work is to present a Constraint Programming (CP) approach to this problem and also four practical situations which demand more sophisticated constraints than just the avoiding of conflicts. These sort of constraints have hitherto been ignored in the literature [7].

The first of them, which is very useful in practical situations, allows the planner to specify a *meeting station* for a pair of trains and a minimum dwell time during which they must be together at the station, for changing either crew or goods between these two trains. The second provides the dispatcher with the possibility of specifying that a particular vehicle must be used to *form* another trip. In this case, the second trip must necessarily be scheduled after the first trip ends, plus the minimum specified time. The third is the *blocking* of a track segment for a certain period of time for maintenance. Finally, the planner can also specify a *time headway* limit. This limit specifies a minimum time which trains must stay apart from each other during the trip either for safety or any other operational reasons. These constraints can be taken into account in the current implementation.

The numerical results show that the CP approach proposed here is a promising alternative to MIP for scheduling real-world instances of single-track railway problem.

This paper is organized as follows. In Section 2 a model is presented which maps the single-track railway scheduling problem into a special case of Job-Shop Scheduling problem.

An outline of the algorithm used to perform the experiments is given in Section 3, and in Section 4 the results of solving 21 problems gathered from Higgins' work [3] is presented. The conclusions are given in Section 5.

2 The Disjunctive Graph Model

The single-track railway can be modelled as a special case of the jobshop scheduling problem. This can be achieved by considering the train trips as jobs, which will be scheduled on tracks regarded as machines. A train trip may consist of many operations that require traversing from one point to another on a track. Each of these distinct points can be a station or a signal placed along the track separating track segments. In order to eliminate a conflict an operation is chosen from the pair of conflicting operations to be delayed up to the point where the conflict is resolved.

The job-shop scheduling problem is a class of combinatorial problem well known in the OR and AI literature and defined as follows. Given are a set of jobs $\mathcal{J}$. For each job $J_i \in \mathcal{J}$ a set of operations $J_i = \{o_{i1}, o_{i2}, \ldots, o_{ik}\}$ is also specified. Each of these tasks requires processing on a unique machine $r_i \in \mathcal{R}$, the set of machines. A machine r_i has capacity > 1 to perform more than one task simultaneously when representing a passing point, and capacity $= 1$ when representing a track segment.

The p_{ij} processing time for each operation o_{ij} is also given as input for the problem. In addition, each job has its release date d_i and its expected completion date c_i. If C_i denotes the actual completion time for job J_i, then the tardiness T_i of job J_i is defined as $max(C_i - c_i, 0)$. The aim is then to minimize the total tardiness given by the expression:

$$D = \sum_{J_i \in \mathcal{J}} T_i; \tag{1}$$

This problem can elegantly be represented by a disjunctive graph [8]. Consider a directed graph $G = (N, A, B)$, where N is the set of all the processing operations o_{ij} of fixed processing time p_{ij} of each J_i to be uninterrupted performed. A is a set of *conjunctive* arcs which represent the sequence of operations on a particular job, whereas B is the set of pairs of *disjunctive* arcs linking tasks being performed on the same machine.

The conjunctive arcs of A are in fact precedence constraints so that, for each pair of successive operations $(o_{ij}, o_{i(j+1)})$ of job J_i, o_{ij} must be performed before $o_{i(j+1)}$.

Each machine which represents a track segment can only perform a unique operation at a time, therefore, every operation which is performed on the same machine is linked by two directed arcs in the set B. The semantic of these *disjunctive* arcs is so that one arc is to set the possible precedence order between two pair of tasks as $(o_{ij} \rightarrow o_{i'j'})$, and another arc is to set the alternative order $(o_{i'j'} \rightarrow o_{ij})$. The two alternative orders are set this way until an order between the tasks on the machine is determined.

The conjunctive arcs $(o_{ij}, o_{i(j+1)}) \in A$ linking two consecutive operations of job J_i have associated to them the processing time p_{ij}, that is the processing time of task o_{ij}. Associated to the arc linking two disjunctive operations, for instance $(o_{ij}, o_{i'j'})$, is p_{ij} and associated to the alternative arc linking the same pair of task $(o_{i'j'}, o_{ij})$ is $p_{i'j'}$, in case $o_{i'j'}$ is instead performed before o_{ij} on the machine.

To the graph G we add a dummy node S as source node. This node represents a dummy task and links to the first task of each job. The processing time associated to the arc between these two tasks is d_i, the release time of job J_i.

Besides the set of conjunctive and disjunctive arcs which exist due to the characteristics of the problem, we may also have in the graph G some other fixed additional arcs introduced by the special constraints described in Section 1.

We have $n = \mid J \mid$ dummy nodes N_i to represent the sink nodes. Each of these nodes denote the tardiness of job J_i. The reason for having n sink nodes rather than one is because we are minimizing the total tardiness instead of a unique maximum completion time, as is usual when the objective function is the *makespan*. Moreover, a change in the order of an operation on a machine can also cause a change in completion time of the other jobs.

In order to have a feasible schedule it is necessary to transform the given disjunctive graph of the given timetable into an acyclic graph. This is performed by choosing an order to those disjunctive activities. This means, in train context, to choose which service is going to use first the track segment in dispute.

Any operation o_{ij} whose the execution time coincides either total or in part with another operation $o_{i'j'}$ being performed on the same track segment, is considered conflicting tasks. To arbitrate an order between any conflicting pair of operations we need to use a dispatch rule to decide which stretch of trip (operation) uses first that track segment, while the other trip is delayed. This way only one of the disjunctive arc is left in the disjunctive graph. The order of unconflicting activities is preserved because they do not have any impact on the total delay (see Equation 1) we want to minimize.

Whatever the criteria chosen to arbitrate the order for the conflicting activities is, once the unique time interval for execution of all the operations in the graph are determined, some operations will have an important role in the graph structure: they will be the *critical operations*. The *critical path* is formed by these particular operations. Given a feasible solution, the activities whose schedule order cannot be changed without also changing the value of the objective function [8] are those of the *critical path*.

A neighbour feasible solution can be obtained by reversing the orientation of an arc between two critical activites being performed on the same machine. In train scheduling context, this means opting a different ordering between two conflicting stretch of trips.

In following section we describe the three-stage algorithm we are proposing.

3 The Three-Stage Algorithm

Our algorithm consists of three stages and is developed with the ILOG Scheduler [4]. In the first stage there is an one-step branch-and-bound just to find an initial solution for the problem. In the second stage there is a local search procedure which uses the critical path of the solution in order to find an improved feasible neighbour. In the third stage there is the branch-and-bound used for the first stage. However, given the current upper-bound, the aim in this stage is to free the local search from a local minimum cost solution by finding an improved solution, if there is one.

Therefore, the algorithm described in this paper takes advantage of the best of the local search strategies and the enumerative strategies. From the first for walking quickly through improved solutions finding tighter bounds for the problem, and from the second for thoroughly searching the search space seeking for the optimal solution.

The devised branch-and-bound algorithm iterates chronologically through the list of conflicting operations. For each conflicting operations it fixes an arc orientation between them so that we can have an acyclic graph (see Section 2) after resolving all the conflicting operations. A solution is found when the algorithm reaches a leaf of the search tree, at this point the cost of the found solution is posted as a new upper-bound for the next solution.

To decide the order between a pair of conflicting operations we use the *Shortest Processing Time* (SPT) dispatch rule [8]. The SPT rule chooses the order between two tasks which yields the least additional total delay.

During the process of resolving conflicting operations new conflicts can be created. These new conflicting operations do not ever precede in time the already resolved conflicts.

The local search we propose here is a hill-climbing strategy based upon the introduction of a *perturbation* to the critical path of a solution. A perturbation to the critical path (see description in Section 2) is provoked by changing the orientation of the arc of a former pair of conflicting operations. This perturbation is carried out to generate a feasible neighbour until a better solution than the current solution is found.

Many variants of local search have been proposed in the last decade for dealing with the job-shop scheduling problem [9]. Usually a local search strategy is used as an alternative to enumerative algorithms due to computational high cost of the complete search. Unlike this usual approach Baptiste *et al.* [1] presented a hybrid approximation and enumerative algorithm to speed up the process of reaching near-optimal solutions.

The structural difference between their approach and ours is that their local search works up to a certain number of iterations without improvement and then stops. The enumerative algorithm is then launched to search for the optimal solution. Our branch-and-bound algorithm is instead used to free the local search from the local minimum, if a better solution can be found, returning the control to the local search afterwards.

As already mentioned, new conflicts might be created due to the modifications carried out as described above. The algorithm deals with the new conflicts, using the chosen dispatch rule, by setting up an orientation between each of the new conflicting pair of activities. When all the remaining conflicts are resolved, the cost of this new solution is then compared against the current upper-bound. An improved solution is taken as the new current solution and its cost is posted as a new upper-bound for the next solution. Another neighbour is tried otherwise.

The hill-climbing algorithm is trapped in a local minimum when all the neighbours have been tried without leading to any improvement. The algorithm then hands the control over to the branch-and-bound algorithm to search for a better solution, if there is one. When an improved solution can be found, the control is then returned to the local search, the optimal solution has been found otherwise.

4 Numerical Results

The dataset used to carry out our experiment is that also used by Higgins [3], which is a real-world sample of problems.

Table 1. Problems' classification according to their number of conflicts.

	Problems			
Number	Total	Inbound	N$^{\underline{o}}$ of Passing points	N$^{\underline{o}}$ Conflicts
20-29	7	4	6	6-11
40-46	9	4	6	8-14
50-51	25	12	12	27,35
60,61	30	15	16	62,69

The program described in Section 3 is used here to solve a set of problems, whose characteristics are shown in Table 1. Within this set of problems there are none of the special constraints we suggested in Section 1, although our program can handle them [7]. Therefore, the aim here is twofold. First, to evaluate the performance of the proposed combined approach on solving this set of real-world problems. Second, to compare the performance of the combined approach against that obtained by the branch-and-bound alone.

In Table 1 the *Number* column gives the problem number, *Total* is the total number of trains to schedule in the problem, *Inbound* is the number of inbound trains, *N$^{\underline{o}}$ of Passing points* states the number of passing points which trains can use to pass each other; and *N$^{\underline{o}}$ of Conflicts* is the range of conflicts which the problems of the group have.

The problems are classified according to the number of conflicts they have in their given original timetable. Higgins suggested that the number of conflicts in a problem is a good measure of its difficulty [3].

We used our combined program to solve the 21 problems described in Table 1 and the result yielded is depicted in Table 2. These experiments were performed on a networked PC Pentium III.

The *First Solution* columns show the time to find the first feasible solution to a problem and its respective delay. Likewise, the *Best Solution* columns show the result to reach the best solution for that particular problem and the overall minimum delay.

The *Iterative Improvement* columns shows the first local minimum solution reached by the local search method proposed here and described in Section 3. The *Time*, in this column, is the necessary time to find the first improved local

Table 2. Results of solving 21 problems by using the combined approch.

	First Solution		Iterative Improvement			Best Solution		
Problem	Time	Delay	Time	Delay	LM	Time	Delay	CPU
20	0.03	1815	0.37	1775	14	5.22	1224	5.27
21	0.03	253	0.14	210	1	0.14	175	0.15
22	0.03	294	0.08	238	2	0.14	137	0.15
23	0.02	116	0.06	151	1	0.06	116	0.07
24	0.03	517	0.12	391	2	0.22	349	0.25
25	0.03	410	0.12	285	1	0.12	285	0.14
26	0.02	344	0.07	352	2	0.13	338	0.15
27	0.02	303	0.09	310	2	0.16	161	0.17
28	0.02	249	0.07	257	1	0.07	249	0.09
29	0.03	409	0.12	321	2	0.23	223	0.24
40	0.02	350	0.09	282	2	0.15	215	0.16
41	0.02	199	0.11	334	1	0.11	199	0.12
42	0.01	441	0.16	439	4	0.51	388	0.58
43	0.03	716	0.15	690	4	0.83	560	1.03
44	0.01	427	0.14	476	2	0.23	389	0.26
45	0.03	359	0.09	291	1	0.09	228	0.11
46	0.02	526	0.12	526	1	0.12	526	0.21
50	0.17	805	3.64	565	2	5.98	555	22.85
51	0.21	1445	18.27	810	6	406.13	650	463.78
60	0.90	504	92.88	318	–	–	–	–
61	1.09	800	51.83	345	–	–	–	–

minimum solution and the *Delay* shows its value. When no improved solution can be found, *i.e.* when the heuristic is trapped, from the start, into a local minimum, the last local minimum cost solution found is shown instead. The *LM* number shows how many times the local search got trapped in a local minimum and the branch-and-bound algorithm was called to free the heuristic from it.

In the *CPU* column is shown the total time in seconds including the time to find the best solution and prove its optimality. The symbol '–' shows when no solution has been found after 15 hours. For instance, the combined algorithm managed to find good solutions for problems 60 and 61 in about 3min. These solutions are not reached by the branch-and-bound when running alone up to 15 hours.

The results of problems 20 and 51 turn our attention to the number of times the iterative improvement strategy was trapped into a local minimum and the correspondent amount of time to reach the best solution, 5.22 and 406.13 respectively. In addition to showing the local search part of the program did not help much the algorithm in walking smoothly through improved solutions for these two problems, these results also show the poor quality of the neighbourhood on yielding better solutions for these particular problems.

For problems 23, 26, 27, 28, 41 and 44, the local search was not able to find a better solution than the initial solution. This is shown in *Iterative Improvement* column in Table 2 by the delay value. However, after the improved solution found by branch-and-bound, the local search managed to find other solutions for problems 26, 27, and 44, this is shown by the fact that the branch-and-bound was called more than once in these cases.

Among these 21 problems, the optimal solution was promptly found for four problems, problems 23, 28, 41 and 46. With the exception of problem 46, the iterative method did not managed to find these optimal solutions from their respectively neighbourhood in order to meet the cost of the first solution.

The combined algorithm proposed here did improve the performance on finding better solutions for those large problems (51, 60, and 61). However, it is still difficult to reach the optimal solution and prove optimality for problems 60 and 61 within a maximum of 15 hours.

5 Conclusions

This paper presents a model which maps the Single-Track Railway scheduling problem to a Job-Shop Scheduling problem. Some practical constraints that arise in train operating context are described and incorporated into the solving program.

The results show that CP is promising alternative to MIP for solving real-life instances of problems, specially because of the possibility of combining different strategies altogether to get the best of them. In addition, some practical constraints which are hard to specify in MIP, are manageable with CP.

References

1. Baptiste, P., Le Pape, C., and Nuijten, W. Constraint-Based Optimization and Approximation for Job-Shop Scheduling. AAAI-SIGMAN Workshop on Intelligent Manufacturing Systems, IJCAI-95, Montreal Canada, 1995.
2. Cai, X. and Goh, C.J. A Fast Heuristic for The Train Scheduling Problem. *Computers and Operations Research*, 21(5):499–510, 1994.
3. Higgins, A., Kozan, E., and Ferreira, L. Optimal Scheduling of Trains on a Single Line Track. *Transportation Research*, 30(2):147–161, 1996.
4. ILOG S.A. ILOG *Scheduler Reference Manual, 4.0 edition 1997*.
5. Jovanović, D. *Improving Railroad On-Time Performance: Models, Algorithms and Applications*. PhD thesis, The Wharton School, University of Pennsylvania, 1989.
6. Kreuger, P., Carlson, M., Olsson, J., Sjöland, T., and Aströ,E. Trips Scheduling on Single Track Networks - The TUFF Train Scheduler. *CP'97 Workshop on Industrial Constraint – Directed Scheduling*, pages 1–12, 1997.
7. Oliveira, E. *Solving Single-Track Railway Scheduling Problem Using Constraint Programming*. PhD thesis, School of Computing, University of Leeds, September 2001.
8. Pinedo, M. *Scheduling : Theory, Algorithms and Systems*. Prentice Hall, 1995.
9. Vaessens, R.J.M., Aarts, E.H.L., and Lenstra, J.K. Job Shop Scheduling by Local Search. *INFORMS Journal on Computing*, 8(3):302–317, 1996.

A Temporal Planning System for Time-Optimal Planning[⋆]

Antonio Garrido, Eva Onaindía, and Federico Barber

Dpto. Sistemas Informáticos y Computación
Universidad Politécnica de Valencia
Camino de Vera s/n, 46022 Valencia, Spain
Fax: (+34) - 963877359
{agarridot,onaindia,fbarber}@dsic.upv.es

Abstract. Dealing with temporality on actions presents an important challenge to AI planning. Unlike Graphplan-based planners which alternate levels of propositions and actions in a regular way, introducing temporality on actions unbalance this symmetry. This paper presents TPSYS, a *Temporal Planning SYStem*, which arises as an attempt to combine the ideas of Graphplan and TGP to solve temporal planning problems more efficiently. TPSYS is based on a three-stage process. The first stage, a preprocessing stage, facilitates the management of constraints on duration of actions. The second stage expands a temporal graph and obtains the set of temporal levels at which propositions and actions appear. The third stage, the plan extraction, obtains the plan of minimal duration by finding a flow of actions through the temporal graph. The experiments show the utility of our system for dealing with temporal planning problems.

1 Introduction

In many real world planning problems, time plays a crucial role. In these problems it is necessary to discard the assumption that actions have the same duration. For instance, it is clear that in a logistics domain the operator $\texttt{fly}-\texttt{plane}$ may be longer than $\texttt{load}-\texttt{plane}$, and the action $\texttt{fly}-\texttt{plane}(\texttt{London},\texttt{Moscow})$ is longer than $\texttt{fly}-\texttt{plane}(\texttt{London},\texttt{Paris})$. Hence, dealing with temporal planning problems requires to handle a set of more complex constraints since the difficulty does not only lie in the process of action selection [1], but also in the process of selecting the right execution times for actions. In addition, the criterion for optimization changes because in temporal contexts the interest lies in obtaining a plan of minimal duration rather than a plan of minimal number of actions. Therefore, finding the plan which minimizes the global duration becomes an important issue in temporal planning.

[⋆] This work has been partially supported by the Project n. 20010017 - *Navigation of Autonomous Mobile Robots* of the Universidad Politécnica de Valencia.

Dealing with temporality on planning has also been tackled by introducing independent scheduling techniques in charge of doing all the reasoning on time and resources. Under this approach of two separated processes, the planner obtains a plan, which is validated afterwards by the scheduler. In case the scheduler determines this plan is unfeasible, the planner must obtain a new plan and the same procedure is repeated again. Obviously, this approach entails a large overhead in the overall process. For that reason, a few attempts at integrating planning and scheduling are being carried out [2]. TPSYS [3] follows the guidelines of these latter approaches by achieving a total integration of time and actions in the same framework.

This paper extends the work in [3] and aims to solve the above drawbacks. It builds on the work of Smith and Weld (the *Temporal Graphplan* algorithm, TGP, presented in [4]) and examines the general question of including temporality on actions in a Graphplan-based approach [5] by guaranteeing the plan of minimal duration. We present a *Temporal Planning SYStem* (TPSYS) which consists of three stages: a *preprocessing* stage, a temporal graph expansion stage and a plan extraction stage. The main features of TPSYS are:

- It is able to handle overlapping actions of different duration and guarantees the optimal plan, i.e. the plan of minimal duration.
- It defines a new classification of mutual exclusion relations: *static* mutexes which are time independent and *dynamic* mutexes which are time dependent and transitory.
- It expands a relaxed temporal graph (from now on *TG*) without maintaining no − op actions nor delete-edges. The *TG* is incrementally expanded through temporal levels defined by the instants of time at which propositions appear.
- It performs a plan extraction (from now on *PE*) stage by selecting the appropriate actions in the *TG* to achieve the problem goals and guaranteeing the plan of minimal duration. Consequently, the algorithm can also solve problems of the type *'obtain a plan of duration shorter than* $\mathcal{D}_{\max}$ *'*.

This paper is organized as follows. Section 2 discusses some related work in temporal planning and briefly introduces the main differences between TGP and TPSYS. The three stages of TPSYS, the necessary definitions and an application example of the proposed system are presented in section 3. Some experimental results, comparing the performance of TGP and TPSYS are shown in section 4. Finally, the conclusions of the work and the future lines on which we are working now are presented in section 5.

2 Related Work

Despite the great effort to introduce temporal constraints and resources in planning, none of the existing works have been largely used. Presumably, the reason is they do not exhibit a good performance when dealing with temporal problems due to the complexity these problems entail. If we focus on the last decade, one of the first temporal planners was O-Plan [6] which integrates both planning and scheduling processes into a single framework. Then, other planners such as ZENO [7] or IxTeT [8] appeared in the literature. Although IxTeT does not manage disjunctive constraints, it deals with resource availability and temporal constraints by using a TCN (*Temporal Constraint Network* [9]) to represent constraints on time points. An alternative system for integrating planning and scheduling is performed in HSTS (*Heuristic Scheduling Testbed System* [10]) which defines an integrated framework to solve planning and scheduling tasks in specific domains of spatial missions. This system uses multi-level heuristic techniques to manage resources under the constraints imposed by the action schedule. The parcPLAN approach [11] manages multiple capacity resources with actions which may overlap, instantiating time points in a similar way to our approach. parcPLAN behaves efficiently if there are enough resources but with stricter resource limitations it becomes more inefficient. Köehler deals with planning under resource constraints by means of time producer/consumer actions but without incorporating an explicit model of time in [12].

TGP [4] introduces a complex mutual exclusion reasoning to handle actions of differing duration in a Graphplan context. TPSYS combines features of both Graphplan and TGP and introduces new aspects to improve performance. While TGP aims to demonstrate that its mutual exclusion reasoning remains valuable in a temporal setting, TPSYS aims to guarantee the optimal plan for any temporal planning problem.

The reasoning on *conditional* mutex (involving time mutex) between actions, propositions and between actions and propositions is managed in TGP by means of inequalities which get complex in some problems. In fact, the application of conditional mutexes imposes a set of sophisticated formulae (even with binary disjunctions) which may imply an intractable reasoning on large problems [4]. On the contrary, the reasoning process in TPSYS is simplified thanks to the incorporation of several improvements:

– Static mutex relations between actions and between actions and propositions are calculated in a preprocessing stage because they only depend on the definition of the actions. This allows us to speed up the process of calculating the dynamic mutex relations between propositions while generating the *TG*, obtaining better execution times than TGP.
– TPSYS uses a multi-level temporal planning graph as Graphplan where each level represents an instant of time. Both TPSYS and TGP build a relaxed graph by ignoring $\texttt{no}-\texttt{op}$ actions and delete-edges which allows an extremely fast graph expansion[1] [13]. While in TGP actions and propositions are only annotated with the first level at which they appear in the planning graph, TPSYS annotates all different instances of actions and propositions produced along time. The compact encoding of TGP reduces vastly the space costs but it increases the complexity of the search process, which may traverse cycles in the planning graph. However, the *PE* in TPSYS is straight-

[1] The *TG* expansion represents a small percentage of the execution time in TPSYS. In general, hundreds of temporal levels can be generated in few seconds for many classical domains.

forward because it merely consists of obtaining the plan as an acyclic *flow* of actions throughout the *TG*. The experiments show that using a much more informed *TG* vastly reduces the overhead during the *PE*, thus obtaining better global execution times than TGP.

In addition to the temporal goals which TGP is able to manage, TPSYS can also manage propositions of the initial situation which hold at other times than $t = 0$.

3 Our Temporal Planning SYStem

In TPSYS, a temporal planning problem is specified as a 4-tuple $\{\mathcal{I}_s, \mathcal{A}, \mathcal{F}_s, \mathcal{D}_{\max}\}$, where $\mathcal{I}_s$ and $\mathcal{F}_s$ represent the initial and final situation respectively, $\mathcal{A}$ represents the set of actions, and $\mathcal{D}_{\max}$ stands for the maximal duration of the plan required by the user. Time is modelled by $\mathbb{R}^+$ and their chronological order. Each action is assigned a nondisjunctive positive duration, which may be different. A temporal proposition is represented by $<\mathtt{p},\mathtt{t}>$ where $\mathtt{p}$ denotes the proposition and $\mathtt{t} \in \mathbb{R}^+$ represents the time at which $\mathtt{p}$ is produced. Hence, $\mathcal{I}_s$ and $\mathcal{F}_s$ are formed by two set of temporal propositions $\{<\mathtt{p_i},\mathtt{t_i}>/\mathtt{t_i} \leq \mathcal{D}_{\max}\}$. It must be noticed that propositions in $\mathcal{I}_s$ ($\mathcal{F}_s$) can be placed (required) at any time $\mathtt{t_i}$ through the execution of the plan.

TPSYS follows three consecutive stages, as shown in Fig. 1. After the first stage, the second and the third stage are executed in an interleaved way until a plan is found or the duration of the plan exceeds $\mathcal{D}_{\max}$.

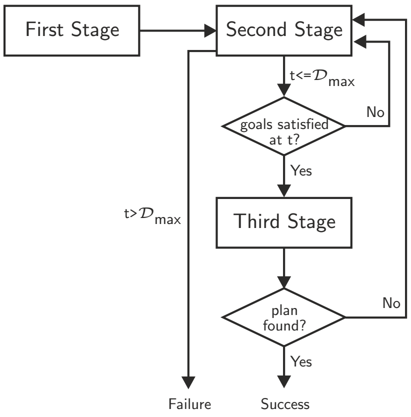


Fig. 1. Flow chart of the three stages of TPSYS

In this section, we will make use of the action domain defined in Table 1 to show the behaviour of our system. Table 1 presents a description of the actions

Table 1. Simplified *Briefcase* domain: only three actions are defined to achieve the goal proposition `at(B1,U)`

Action	Duration	Precs.	Effects

`ld(B1,BC,H)`	5	`at(B1,H)`
		`at(BC,H)`
		`free(BC)`
			`in(B1,BC)`
			`¬at(B1,H)`
			`¬free(BC)`

`mv(BC,H,U)`	5	`at(BC,H)`	`at(BC,U)`
			`¬at(BC,H)`

`uld(B1,BC,U)`	2	`in(B1,BC)`
		`at(BC,U)`
			`at(B1,U)`
			`free(BC)`
			`¬in(B1,BC)`

(with duration) of the well-known *Briefcase* domain. In order to simplify the domain only three actions are defined, those which are necessary to transport a book (`B1`) from home (`H`) to university (`U`) by using a briefcase (`BC`).

3.1 First Stage: Preprocessing

At this preprocessing stage, TPSYS calculates the static mutual exclusions which will facilitate the computation of dynamic mutex relations during the second stage. A mutex relationship between actions is defined as follows [5]: two actions `a` and `b` are mutex if `a` deletes a precondition of `b`, or `a` and `b` have conflicting effects, in whose case they cannot be executed in parallel. Mutex between propositions appears as a consequence of mutex between actions. Thus, two propositions `p` and `q` are mutex if all actions for obtaining `p` are mutex with all actions for obtaining `q`.

Let `a` and `b` be two actions, and let S_1, $S_2 \in \{\texttt{precs(a)}, \texttt{add}-\texttt{effs(a)}, \texttt{del}-\texttt{effs(a)}, \texttt{precs(b)}, \texttt{add}-\texttt{effs(b)}, \texttt{del}-\texttt{effs(b)}\}/S_1 \neq S_2$. We define the function $\texttt{coincidences}(S_1, S_2)$ as a boolean function which holds iff $\exists \texttt{p}/\texttt{p} \in S_1 \wedge \texttt{p} \in S_2$. According to this, we introduce the following definitions:

Definition 1. ***Static mutex between actions.*** *Actions* `a` *and* `b` *are statically mutex if they cannot be executed in parallel. Although this relationship is equivalent to Graphplan's interference, we break it down into two different types of mutex relationships. Hence, if two statically mutex actions are given at the same instant of time, these actions will have to be executed in one of the following ways:*

a) ***Consecutive actions****, iff*
$\texttt{coincidences}(\texttt{add}-\texttt{effs(a)}, \texttt{del}-\texttt{effs(b)}) \vee$
$\texttt{coincidences}(\texttt{add}-\texttt{effs(b)}, \texttt{del}-\texttt{effs(a)})$.

Clearly, if `a` *and* `b` *have conflicting effects, the correct execution order is '*`a after b`*' or '*`b after a`*'. In Table 1, actions* `ld(B1,BC,H)` *and* `uld(B1,BC,U)` *can* *be executed in any consecutive order because their only interdependency is the conflicting effect* `in(B1,BC)`*.*

***b) Consecutive actions in an after order**, iff*
$\texttt{coincidences}(\texttt{del}-\texttt{effs}(\texttt{a}), \texttt{precs}(\texttt{b}))\wedge$
$\neg\texttt{coincidences}(\texttt{precs}(\texttt{a}), \texttt{del}-\texttt{effs}(\texttt{b}))$.

Clearly, if `a` *deletes a precondition of* `b` *but* `b` *does not delete any precondition of* `a`*, the correct execution order is '*`a after b`*'. In Table 1, action* `mv(BC,H,U)` *must be executed after* `ld(B1,BC,H)` *because of the proposition* `at(BC,H)`*.*

Definition 2. ***Static ap-mutex (static action/proposition mutex).*** *One action* `a` *is statically ap-mutex with a proposition* `p` *iff* $\texttt{p} \in \texttt{del}-\texttt{effs}(\texttt{a})$*. For instance,* `ld(B1,BC,H)` *is ap-mutex with* `at(B1,H)` *and* `free(BC)` *in Table 1.*

3.2 Second Stage: Temporal Graph Expansion

We adopt the same conservative model of action as in TGP, in which i) all preconditions must hold at the start of the execution of the action, ii) preconditions *not deleted* by the action itself must hold during the entire execution of the action, iii) effects are undefined during the execution and only guaranteed to hold at the end of the action, and additionally, iv) actions can start their execution as soon as all their preconditions hold.

Previously to start with the process performed in the second stage, we introduce the following definitions to better understand the expansion of the temporal graph.

Definition 3. ***Temporal graph (TG).*** *A TG is a directed, layered graph with two types of nodes (proposition and action nodes) and two types of edges (precondition-edges and add-edges). The TG alternates proposition and action levels like Graphplan. Each level is labelled with a number representing the instant of time at which propositions are present and actions start their execution. Levels are ordered by their instant of time. This way, the algorithm can easily move from a level t to the next level $t' > t$ during the TG expansion, and to the previous level $t'' < t$ during the plan extraction. The TG is expanded by a forward-chaining process which simply adds the add-effects of actions (delete-effects are ignored) at the proper level according to the duration of actions.*

Definition 4. ***Instance of an action.*** *We define an instance of an action* `a` *as the triple* $<\texttt{a},\texttt{s},\texttt{e}>$ *where* `a` *denotes the action and* $\texttt{s},\texttt{e} \in \mathbb{R}^+$ *represent the time when the instance starts and ends executing, respectively (*$\texttt{e} = \texttt{s} + duration(\texttt{a})$*).*

Definition 5. ***Proposition level.*** *A proposition level $P_{[t]}$ is formed by the set of temporal propositions* $\{<\texttt{p}_\texttt{i},\texttt{t}_\texttt{i}>/\texttt{t}_\texttt{i} \leq \texttt{t}\}$ *present at time* `t` *which verify* $<\texttt{p}_\texttt{i},\texttt{t}_\texttt{i}> \in \mathcal{I}_s \vee \exists <\texttt{a}_\texttt{i},\texttt{s}_\texttt{i},\texttt{e}_\texttt{i}>/\texttt{p}_\texttt{i} \in \texttt{add}-\texttt{effs}(\texttt{a}_\texttt{i}), \texttt{e}_\texttt{i} = \texttt{t}_\texttt{i}$*. It must be noticed that '$<$' between* $\texttt{t}_\texttt{i}$ *and* `t` *is used to denote* $\texttt{p}_\texttt{i}$ *persists in time. Consequently, if a proposition is present at $P_{[t]}$, it will appear at all $P_{[t']}$ such that* $\texttt{t}' > \texttt{t}$*.*

Definition 6. ***Dynamic mutex between temporal propositions at*** $P_{[t]}$**.** *Let* $\{<\mathtt{a_i},\mathtt{s_i},\mathtt{t_i}>\}$ *and* $\{<\mathtt{b_j},\mathtt{s_j},\mathtt{t_j}>\}$ *be two sets of instances of actions which achieve* $<\mathtt{p},\mathtt{t_i}>,<\mathtt{q},\mathtt{t_j}> \in P_{[t]}$ *respectively (i.e.,* $\mathtt{p} \in \mathtt{add}-\mathtt{effs}(\mathtt{a_i})$ *and* $\mathtt{q} \in \mathtt{add}-\mathtt{effs}(\mathtt{b_j})$*). Temporal propositions* $<\mathtt{p},\mathtt{t_i}>$ *and* $<\mathtt{q},\mathtt{t_j}>$ *are dynamically mutex at* $P_{[t]}$ *iff i)* $\forall\alpha,\beta/\alpha \in \{<\mathtt{a_i},\mathtt{s_i},\mathtt{t_i}>\}, \beta \in \{<\mathtt{b_j},\mathtt{s_j},\mathtt{t_j}>\}$, α *and* β *overlap and ii)* $\mathtt{a_i}$ *and* $\mathtt{b_j}$ *are statically mutex. Otherwise, temporal propositions are not dynamically mutex at* $P_{[t]}$*. Loosely speaking, two propositions are dynamically mutex at* $P_{[t]}$ *if they cannot be simultaneously available at time* $\mathtt{t}$*. Obviously, a dynamic mutex may expire as new levels are expanded further in the TG, and the involved statically mutex actions are ordered in sequence.*

Definition 7. ***Action level.*** *An action level* $A_{[t]}$ *is formed by the set of instances of actions* $\{<\mathtt{a_i},\mathtt{t},\mathtt{e_i}>\}$ *which start their execution at time* $\mathtt{t}$ *because* $\forall<\mathtt{p},\mathtt{t_i}>,<\mathtt{q},\mathtt{t_j}>\in P_{[t]}/\mathtt{p},\mathtt{q} \in \mathtt{precs}(\mathtt{a_i})$, $<\mathtt{p},\mathtt{t_i}>,<\mathtt{q},\mathtt{t_j}>$ *are not dynamically mutex at* $P_{[t]}$*.*

Proposition 1. *Let* $P_{[t]}$ $(t \leq \mathcal{D}_{\max})$ *be the earliest proposition level at which all temporal propositions in* $\mathcal{F}_s$ *are pairwise not dynamically mutex. Under this assumption, no correct plan can be found before time* t *(i.e., the duration of a correct plan will never be shorter than* t*).*

Proof. According to Definition 6 the proof is simple: if each pair of goal propositions are not simultaneously available until time t, the minimal duration as possible of a plan will be t. Notice, however, that this statement does not guarantee that the minimal duration of a correct plan is always t because delete-effects have been ignored during the *TG* expansion and, consequently, some goal propositions might have been deleted.

The second stage of TPSYS expands the *TG* (see Fig. 2) by alternating proposition and action levels through a fast forward-chaining process. The *TG* expansion differs from Graphplan in the following points:

- $\mathtt{No}-\mathtt{op}$ actions are never generated because propositions persist in time through proposition levels.
- Each instance of an action $<\mathtt{a},\mathtt{s},\mathtt{e}>$ in $A_{[s]}$ does not add its add-effects into $P_{[s+1]}$ but in $P_{[e]}$. Thus, different proposition levels are now generated from a given action level.
- The only mutual exclusion relationship checked during the *TG* expansion is the dynamic mutex relationship between temporal propositions.

Starting at $P_{[0]}$ (we will suppose there exists at least one temporal proposition in $\mathcal{I}_s$ which belongs to $P_{[0]}$), the algorithm moves incrementally in time (from one level to the next) throughout the *TG* generating new action and proposition levels. At each action level $A_{[t]}$, the algorithm generates the entire set of instances of actions which start their execution because their preconditions are not dynamically mutex at $P_{[t]}$. After generating each instance of an action, the

```
Algorithm TG expansion
$\forall$ <$\mathtt{p_i}$,$\mathtt{t_i}$> $\in \mathcal{I}_s$ do initialize $P_{[\mathtt{t_i}]}$
$t = 0$
while $(t \leq \mathcal{D}_{\max}) \wedge (\mathcal{F}_s$ is not satisfied in $P_{[t]})$ do
    $\forall\ \alpha =$<$\mathtt{a_i}$,$t$,$\mathtt{e_i}$> which starts at $A_{[t]}$ do
        $A_{[t]} = A_{[t]} \cup \alpha$
        $P_{[\mathtt{e_i}]} = P_{[\mathtt{e_i}]} \cup \mathtt{add} - \mathtt{effs}(\mathtt{a_i})$
    $t =$ next level in the TG
endwhile
```

Fig. 2. Algorithm for the *TG* expansion

propositions in add − effs are added into the proper proposition level. The *TG* expansion terminates once all temporal propositions in the final situation are present in $P_{[t]}$ and none are pairwise dynamically mutex (i.e. $\mathcal{F}_s$ is satisfied in $P_{[t]}$). Moreover, if $t > \mathcal{D}_{\max}$ the algorithm outputs '*Failure*' because no feasible plan can be found earlier than $\mathcal{D}_{\max}$.

The resulting *TG* for the domain defined in Table 1 is shown in Fig. 3. Action uld(B1,BC,U) cannot start at $A_{[5]}$ because its preconditions in(B1,BC) and at(BC,U) are dynamically mutex at $P_{[5]}$ and they cannot be simultaneously available until $P_{[10]}$. Goal proposition at(B1,U) is achieved at $P_{[12]}$ and, therefore, the *TG* expansion terminates at that level.

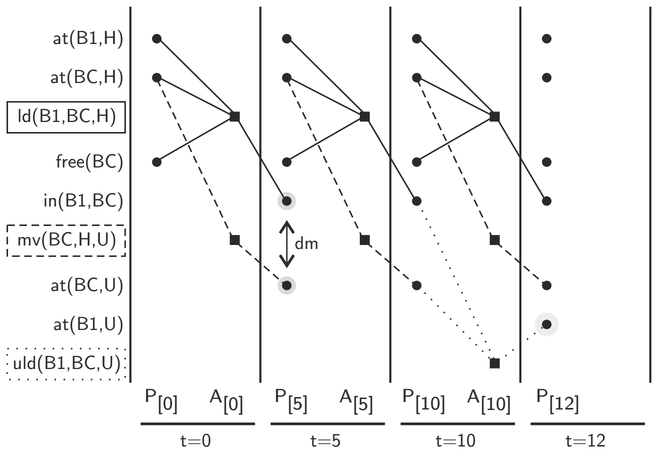


Fig. 3. *Temporal Graph* for the *Briefcase* problem defined in Table 1

3.3 Third Stage: Plan Extraction

The third stage is a backward search process throughout the *TG* to extract a feasible plan. The algorithm uses two data structures `PlannedActs` and `GoalsToSatisfy` which are indexed by a level. `PlannedActs`, which is initialized empty, stores the instances of actions planned at each action level. `GoalsToSatisfy` stores the temporal propositions to be satisfied at each proposition level, and it is initialized by inserting all the temporal propositions in $\mathcal{F}_s$.

Assuming the *PE* process starts from the proposition level $P_{[t]}$ (that is, the search starts from time t, which indicates the duration of the plan to be extracted, in the *TG*), where all temporal goals in $\mathcal{F}_s$ are not dynamically mutex, the algorithm proceeds in the following way:

1. If $t = 0$ and $\texttt{GoalsToSatisfy}[t] \not\subseteq \mathcal{I}_s$, then fail (backtrack) —this is the base case for the recursive process.
2. If $\texttt{GoalsToSatisfy}[t] = \phi$ then move backwards in time (t =previous level in the *TG*) and go to step 1 to satisfy the goals at t.
3. Extract a temporal proposition $<\texttt{p}, t>$ from $\texttt{GoalsToSatisfy}[t]$.
4. Select an instance of an action[2] $\alpha =<\texttt{a}_\texttt{i}, \texttt{s}_\texttt{i}, \texttt{e}_\texttt{i}>/\texttt{p} \in \texttt{add} - \texttt{effs}(\texttt{a}_\texttt{i}), \texttt{e}_\texttt{i} \leq t$ (*backtracking point* to guarantee completeness). In order to guarantee the correctness of the plan, α is discarded if at least one of the following conditions holds:
   - $\exists\beta =<\texttt{b}_\texttt{j}, \texttt{s}_\texttt{j}, \texttt{e}_\texttt{j}>\in \texttt{PlannedActs}/\alpha$ and β overlap and $\texttt{a}_\texttt{i}$ and $\texttt{b}_\texttt{j}$ are statically mutex.
   - $\exists<\texttt{q}, \texttt{e}_\texttt{i}>\in \texttt{GoalsToSatisfy}/\texttt{a}_\texttt{i}$ is statically *ap-mutex* with `q` (i.e. α deletes `q`).

   If α is discarded, another instance of an action is selected by *backtracking* to step 4. Otherwise, `p` is satisfied and the structures $\texttt{PlannedActs}[\texttt{s}_\texttt{i}]$ and $\texttt{GoalsToSatisfy}[\texttt{s}_\texttt{i}]$ are updated with α and $\texttt{precs}(\texttt{a}_\texttt{i})$ respectively. Then, the algorithm goes to step 2 to satisfy another (sub)goal.

Since delete-effects have been ignored during the *TG* expansion, it may be not possible to achieve the problem goals from the proposition level $P_{[t]}$. Thus, if the *PE* process does not find a feasible plan from the proposition level $P_{[t]}$, TPSYS backs to the second stage to continue extending more levels before executing the third stage again (see Fig. 1).

Proposition 2. ***TPSYS is complete.***

Informally, we can show TPSYS is complete. In TPSYS, all levels at which propositions and actions may appear are generated during the *TG* expansion. It is clear that, if a solution plan exists for the problem, this plan will be found in the *TG* comprised from $P_{[0]}$ to one of the generated proposition levels and it cannot be present in a proposition level which does not appear in the *TG*.

[2] In TPSYS, all instances of actions α are executed as late as possible, unless this leads to an unfeasible plan.

Additionally, when the *PE* process searches for a correct plan, all instances of actions which achieve the (sub)goals are considered. Therefore, if a solution exists, TPSYS finds it.

Proposition 3. ***TPSYS is optimal.***

Now, we will demonstrate TPSYS is optimal. Let us suppose the *PE* process finds a plan of duration t (extracted from level $P_{[t]}$) represented by $\mathcal{P}_t$ which is non-optimal. If $\mathcal{P}_t$ is non-optimal it implies that $\exists \mathcal{P}'_{t'}$ of duration $t' < t$ (extracted from level $P_{[t']}$) which has not been found by the *PE* process. But, since $\mathcal{P}'_{t'}$ is a correct plan all the temporal goals in $\mathcal{F}_s$ would be not dynamically mutex at proposition level $P_{[t']}$, and the second stage would have terminated at time t. According to Proposition 2, TPSYS is complete and the *PE* process would have found the plan $\mathcal{P}'_{t'}$, which implies that level $P_{[t]}$ would have never been generated and, consequently, $\mathcal{P}_t$ would have never been found which is inconsistent with the initial supposition. Consequently, the first plan TPSYS finds is the optimal plan.

Intuitively, the optimality of TPSYS is clear because it is based on Graphplan. Graphplan obtains the plan with the minimal number of planning steps (levels), because it moves throughout the planning graph and it starts the *complete* extraction of a plan first from the levels with minimal number of planning steps. TPSYS behaves in a similar way, moving throughout the *TG* and extracting the plan first from the earliest temporal levels. This way, when a feasible plan is found the algorithm can guarantee this plan is the plan of minimal duration because the level $P_{[t]}$ of duration t from which the process has started is the earliest level at which the actions of the plan satisfy the problem goals and those actions are not mutex.

3.4 Application Example

Here, we study a simple application example to illustrate the behaviour of our system. This abstract example shows how the proposition and action levels of the *TG* are generated. The description of the actions and their static mutex are shown in Table 2. $\mathcal{I}_s = \{$<p,0>,<q,10>$\}$, so proposition q is not available until time 10. $\mathcal{F}_s = \{$<s,50>,<t,50>,<u,50>$\}$ which implies all the goals must be satisfied no later than 50 ($\mathcal{D}_{\max} = 50$).

An outline with the basic information of the *TG* is shown in Table 3. In this table, we can see the proposition and action levels generated. Although proposition levels have been generated until $P_{[50]}$, the algorithm terminates the *TG* expansion at $P_{[35]}$ because all temporal goals in $\mathcal{F}_s$ are not dynamically mutex at that level. During the *PE* stage, the algorithm selects the instances of actions which obtain these goals, then the preconditions of these actions and so on. The plan obtained by TPSYS is shown in Fig. 4. The optimal duration of the plan is 35 and therefore, if the user constrains the maximal duration to $\mathcal{D}_{\max} < 35$, the algorithm will output '*Failure*' because no plan exists.

Table 2. Domain of the actions in the application example

Action	Duration	Precs.	Effects	*Static mutex*
a	10	p,q	r	d
b	5	r	s	c
c	20	p	¬s,t	b,d
d	15	r	¬p,u	a,c

Table 3. Outline of the *TG* generated by the algorithm

Temporal level t	$P_{[t]}$	$A_{[t]}$
0	p	c
10	p,q	a,c
20	p,q,r,t	a,b,c,d
25	p,q,r,s,t	a,b,c,d
30	p,q,r,s,t	a,b,c,d
35	p,q,r,s,t,u	-
40	p,q,r,s,t,u	-
45	p,q,r,s,t,u	-
50	p,q,r,s,t,u	-

4 Some Experimental Results

To date, our work has been mainly focused on the validation of our system, rather than on code optimization. Based on our experience, TPSYS seems quite promising to deal with temporal planning problems. Although comparison between our approach and other planning systems is quite difficult because they are based on different algorithms, we made a comparison between TPSYS and TGP. The experiments were performed in a 64 Mb. memory Celeron 400 MHz. We solved each problem (some simple, typical *benchmarks* used in AIPS 1998 competition and some examples provided by TGP[3]) with TPSYS and TGP. We can see the results in Table 4 and how the performance of TPSYS is better than TGP in these problems (even in all the examples provided by TGP). Moreover, TPSYS was able to solve problems which TGP was unable, such as `sussman-anomaly` and `briefcase-2b-1c` (which must transport two books with only one container).

As in parcPLAN [11], the main drawback of TPSYS appears in problems with limited resources, such as `briefcase-2b-1c`. In this case it is necessary to plan additional actions to release the used resources and to make them available again. In these problems, the goals are not dynamically mutex at a proposition level $P_{[t]}$, but a valid plan is not found at time t. Unfortunately, dynamic mutex

[3] The source code of TGP and the examples it provides have been obtained at ftp://ftp.cs.washington.edu/pub/ai/tgp.tgz

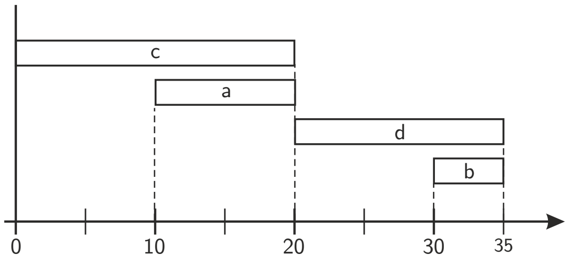


Fig. 4. Optimal plan found by TPSYS for the application example

Table 4. Results of comparison between TPSYS and TGP (times are in milliseconds)

Problem	TPSYS	TGP
tower2	4	40
tower3	8	241
gripper2	4	40
sussman-anomaly	339	-
att-log0	39	40
att-log1	45	200
briefcase-2b-2c	5	251
briefcase-2b-1c	998	-
tgp-AB-p	4	50
tgp-AB-q	4	60
tgp-AB-pq	5	90
tgp-AC-r	4	80
tgp-AC-pr	5	80
tgp-ABDE-r	4	70

relationships are not always enough to determine a proper proposition level from which a feasible plan can be found.

5 Conclusions and Future Work

In this paper we have presented TPSYS, a system for dealing with temporal planning problems. TPSYS arises as a combination of the ideas of Graphplan and TGP to solve temporal planning problems, finding optimal plans and avoiding the complex reasoning performed in TGP.

TPSYS contributes on a classification into static and dynamic mutual exclusion relations. This allows to perform a preprocessing stage which calculates static mutexes between actions and between actions and propositions to speed up the following stages. The second stage expands a *TG* with features of both Graphplan and TGP planning graphs. The third stage guarantees that the first found plan has the minimal duration. Our experiences and the obtained results show the appropriateness of TPSYS for temporal planning problems.

As commented above, TGP presents a mutual exclusion reasoning very valuable for dealing with actions and propositions on a Graphplan-based approach, but unfortunately this complex process may make large problems become intractable. In fact, in our tests TGP was unable to solve a simple problem on the *Briefcase* domain with two books to transport and only one briefcase (which TPSYS was able to solve). Unfortunately, performance of TPSYS degrades in problems with limited resources.

The presented work constitutes a first step towards an integrated system for planning and scheduling. Such a system will be able to manage temporal constraints on actions and to reason on shared resource utilization. Additionally, the system will apply several optimization criteria to obtain the plan of minimal duration or the plan of minimal cost.

Another objective of our current research is the inclusion of resource abstraction as proposed by Srivastava in [14]. Under this proposal, the *TG* will be generated assuming abstract resources, which are always available, avoiding instantiation over particular resources and consequently reducing the size of the *TG*. A special module for reasoning about resources will be included into the *PE* stage to select the specific resource for each action to be planned.

References

1. Smith, D., Frank, J., Jónsson, A.: Bridging the gap between planning and scheduling. Knowledge Engineering Review **15(1)** (2000)
2. Garrido, A., Barber, F.: Integrating planning and scheduling. Applied Artificial Intelligence **15(5)** (2001) 471–491
3. Garrido, A., Onaindía, E., Barber, F.: Time-optimal planning in temporal problems. In: Proc. European Conference on Planning (ECP-01). (2001)
4. Smith, D., Weld, D.: Temporal planning with mutual exclusion reasoning. In: Proc. 16th Int. Joint Conf. on AI (IJCAI-99). (1999) 326–337
5. Blum, A., Furst, M.: Fast planning through planning graph analysis. Artificial Intelligence **90** (1997) 281–300
6. Currie, K., Tate, A.: O-plan: the open planning architecture. Artificial Intelligence **52(1)** (1991) 49–86
7. Penberthy, J., Weld, D.: Temporal planning with continuous change. Proc. 12th Nat. Conf. on AI (1994)
8. Ghallab, M., Laruelle, H.: Representation and control in IxTeT, a temporal planner. In: Proc. 2nd Int. Conf. on AI Planning Systems, Hammond (1994) 61–67
9. Dechter, R., Meiri, I., Pearl, J.: Temporal constraint networks. Artificial Intelligence **49** (1991) 61–95
10. Muscettola, N.: HSTS: Integrating planning and scheduling. In Zweben, M., Fox, M., eds.: Intelligent Scheduling. Morgan Kaufmann, San Mateo, CA (1994) 169–212
11. El-Kholy, A., Richards, B.: Temporal and resource reasoning in planning: the parcPLAN approach. In: Proc. 12th European Conference on Artificial Intelligence (ECAI-96). (1996) 614–618

12. Köehler, J.: Planning under resource constraints. In: Proc. 15th European Conf. on AI (ECAI-98). (1998) 489–493
13. Hoffmann, J., Nebel, B.: The FF planning system: Fast plan generation through heuristic search. Journal of Artificial Intelligence Research **14** (2001) 253–302
14. Srivastava, B., Kambhampati, S.: Scaling up planning by teasing out resource scheduling. In Biundo, S., Fox, M., eds.: Proc. European Conference of Planning (ECP-99), Springer (1999) 172–186

SimPlanner: An Execution-Monitoring System for Replanning in Dynamic Worlds

Eva Onaindia, Oscar Sapena, Laura Sebastia, and Eliseo Marzal

Dpto. Sistemas Informaticos y Computacion
Universidad Politecnica de Valencia
{onaindia,osapena,lstarin,emarzal}@dsic.upv.es

Abstract. In this paper we present SimPlanner, an integrated planning and execution-monitoring system. SimPlanner allows the user to monitor the execution of a plan, interrupt this monitoring process to introduce new information from the world and repair the plan to get it adapted to the new situation.

1 Introduction

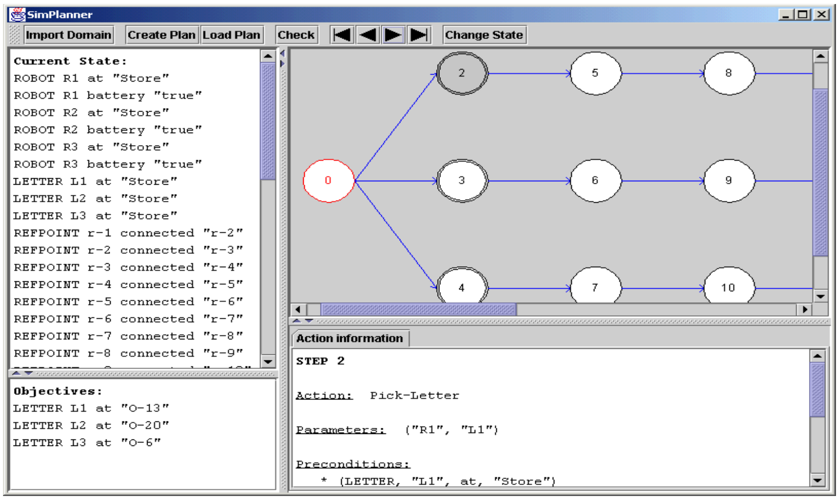
Research on AI planning usually works under the asumption that the world is accessible, static and deterministic. However, in dynamic domains things do not always proceed as planned. Interleaving planning and executing brings many benefits as to be able to start the plan execution before it is completed or to incorporate information from the external world into the planner [8]. Recent works on this field analyse the combination of an execution system with techniques as plan synthesis or anticipated planning [3]. Other works on reactive planning [10] are more concerned with the design of planning arquitectures rather than exploiting the capabilities of the replanning process [9].

SimPlanner is a planning simulator that allows the user to monitor the execution of a plan and introduce/delete information at any time during execution to emulate an external event. It is a domain-independent, synchronous replanner that, like other planning systems [10], avoids generating a complete plan each time by retaining as much of the original plan as possible. SimPlanner has been specially designed for replanning in STRIPS-like domains and has been successfully applied to a great variety of different domains. Additionally, the replanner obtains the optimal solution with respect to the number of actions for most of the test cases.

2 Monitoring the Execution of a Plan

Replanning is introduced during plan execution when an unexpected event occurs in the world. One problem with unexpected effects is deciding how they interact with the effects of the action that was currently being executed. Our solution is to assume the action took place as expected and simply to insert the unexpected effects after the execution of the action.

When an event is produced it is necessary to verify the overall plan is still executable. Many replanning systems only perform a precondition checking to verify whether next action is executable. This option is less costly and much easier to implement in many real applications where the sensory system does not capture all predicates that have changed in the problem. However, this apparently efficient approach may turn out to be inefficient in the long term as many unnecessary actions might be introduced due to changes in the plan are not foreseen enough time in advance.



```
Current State:
ROBOT R1 at "Store"
ROBOT R1 battery "true"
ROBOT R2 at "Store"
ROBOT R2 battery "true"
ROBOT R3 at "Store"
ROBOT R3 battery "true"
LETTER L1 at "Store"
LETTER L2 at "Store"
LETTER L3 at "Store"
REFPOINT r-1 connected "r-2"
REFPOINT r-2 connected "r-3"
REFPOINT r-3 connected "r-4"
REFPOINT r-4 connected "r-5"
REFPOINT r-5 connected "r-6"
REFPOINT r-6 connected "r-7"
REFPOINT r-7 connected "r-8"
REFPOINT r-8 connected "r-9"

Objectives:
LETTER L1 at "O-13"
LETTER L2 at "O-20"
LETTER L3 at "O-6"
```

```
STEP 2

Action:  Pick-Letter

Parameters:  ("R1", "L1")

Preconditions:
  * (LETTER, "L1", at, "Store")
```

Fig. 1. SimPlanner main interface window

Figure 1 shows the graphical interface to monitor a plan execution. The problem corresponds to one of the instances in the robot domain which SimPlanner has been tested on. In the left upper part of the screen it is shown the literals of the current state of the execution. On the right side of the screen a graph representing the plan under execution is displayed. The circles stand for the actions in the plan. Those actions ready to be executed at the next time step are double-circled.

3 Replanning During Execution

The replanning algorithm starts from the current state in the plan execution, when the unexpected event has been input. The objective is to discover which state should be reached next in the problem so as to retain as much of the original plan as is reasonable without compromising the optimal solution. The following two sections explain the algorithm in detail and provide an example to clarify SimPlanner behaviour.

3.1 SimPlanner Algorithm

Initially, the user is monitoring the execution of a plan $P = (a_0 \rightarrow \cdots \rightarrow a_n)$ when an expected event is introduced in the system. The remaining plan to be executed is defined as $\Pi = (a_0\prime \rightarrow \cdots \rightarrow a_n)/\forall a_i \in \Pi \rightarrow a_i \in P$ and a_i has not been executed yet. $a_0\prime$ represents the new current situation (an initial action with effects and no preconditions).

```
Algorithm SimPlanner ($\Pi$) $\rightarrow$ plan $\Pi'$
________________________________________________________________
1. Build a Problem Graph (PG) alternating literal levels and action
levels ($L_0, A_1, L_1, A_2, ...$), where:

        $A_j = \{a_j : \mathsf{Pre}(a_j) \in L_{j-1} \wedge a_j \notin A_i, i < j\}$
        $L_j = L_{j-1} \cup \mathsf{Add}(a)\ \forall a \in A_j$

2. Compute the necessary state, $S(a)$, for each $a \in \Pi$

$S(a_i) = \begin{cases} \{l : l \in \mathsf{Pre}(a_n)\}, i = n \\ \bigcup_{a_j \in \mathsf{Next}(a_i)} S(a_j) \bigcup (\mathsf{Pre}(a_i) - \mathsf{Add}(a_i)) - \bigcup_{a_k \in \mathsf{Paral}(a_i)} \mathsf{Add}(a_k), \text{ otherwise} \end{cases}$

where:
        $\mathsf{Next}(a_i) = \{a_j \in \Pi \wedge a_i \rightarrow a_j\}$
        $\mathsf{Succ}(a_i) = \bigcup_j a_j \in \Pi / a_i \rightarrow \cdots \rightarrow a_j$
        $\mathsf{Paral}(a_i) = \{a_j \in \Pi \wedge a_j \notin \mathsf{Succ}(a_i) \wedge a_i \notin \mathsf{Succ}(a_j)\}$

3. Compute the set of possible reachable states $RS = \{S(c_1), \ldots, S(c_n)\}$,
where:

        $c_i = \{a_1, a_2, \ldots\} / \forall p \in \mathsf{Path}(\Pi) \rightarrow \exists!\ a_k \in p \wedge a_k \in c_i$

$\mathsf{Path}(\Pi)$ the set of all possible paths between $a_0\prime$ and $a_n$, and $S(c_i) =
\bigcup_{\forall a_j \in c_i} S(a_j)$.

4. Select the optimal reachable state $S_{opt} = argmin\{f(x) : x \in RS\}$, where
$f(x) = g(x) + h(x)$.

5. Construction of the final plan $\Pi' = (a_0\prime \rightarrow \cdots \rightarrow S_{opt}) \otimes (S_{opt} \rightarrow \cdots \rightarrow a_n)$.
```

Fig. 2. SimPlanner algorithm

Problem Graph (PG). The first step of the algorithm is to build the PG, a graph inspired in a Graphplan-like expansion [2]. The PG may partially or totally encode the planning problem. The PG is a relaxed graph (delete effects are not considered) which alternates literal levels containing literal nodes and action levels containing action nodes.

The first level in the PG is the literal-level L_0 and it is formed by all literals in the initial situation a_0'. The PG creation terminates when a literal level containing all of the literals from the goal situation is reached in the graph or when no new actions can be applied. This type of relaxed graph is commonly used in many heuristic search planners [4][5] as it allows to easily extract an approximate plan.

Necessary states. Second step is to compute the necessary state to execute each action in Π. A necessary state for an action a_i is the set of literals required to execute a_i and all its sucessors ($\mathsf{Succ}(a_i)$). In order to compute the necessary states, literals are propagated from the goal a_n to the corresponding action by means of the recursive formula shown in the algorithm.

Set of possible reachable states. This set comprises the necessary states to execute a set of parallel actions $\{a_i\}$ and all their successors $\mathsf{Succ}\{a_i\}$. A state will be definitely reachable if its literals make up a feasible situation in the new problem.

In a totally sequential plan Π, the possible reachable states will coincide with the necessary states to execute each action in Π. However, when Π comprises parallel actions it is necessary to compute the combinations of parallel actions, c_i, such that each element in c_i belongs to a different path from the current state to the goal state. In other words, c_i is a set of actions that can all be executed at the same time step. Consequently, the necessary state to execute actions in c_i denotes a state possibly reachable from ${a_0}'$.

Optimal state. In order to select the optimal reachable state, we define a heuristic function $f(x) = g(x) + h(x)$ associated to the cost of a minimal plan from the current situation ${a_0}'$ to the goal state a_n over all paths that are constrained to go through node x. $g(x)$ is the cost of the plan from x to a_n and is calculated straightforward from the original plan. $h(x)$ is the estimated length of an approximate plan P' from ${a_0}'$ to x.

```
Algorithm Approximated plan (${a_0}'$, $x$) $\rightarrow$ plan $P'$
1. Build a ficticious action ${a_n}'$ with preconds and no effects associated
to state $x$
2. $P' = {a_0}' \rightarrow {a_n}'$
2. $L = x - \mathsf{Add}({a_0}')$.
3. while $L \neq 0$
   3.1 select $l \in L$
   3.2 select best $a$ for $l$
   3.3 insert $a$ in $P'$
   3.4 update $L$
4. return $P'$
```

Fig. 3. Outline of the algorithm to build an approximate plan

Step 3.1. Firstly, we form a set $UP \subset L$ with the unsolved preconditions of the nearest action a to ${a_0}'$. If $|UP| > 1$ then literals which appear later in the PG are removed from UP. If again $|UP| > 1$ we count the number of ways for solving each $l \in UP$ ($\{a \in A_i : l \in L_j \wedge l \in \mathsf{Add}(a) \wedge i <= j\}$), and select the literal with the lowest number of actions.

Step 3.2. The best action for $l \in \mathsf{Pre}(a)$ will be the action a_j which minimizes the number of *flaws* (preconditions not yet solved or preconditions of other actions which are deleted by a_j). To compute the number of flaws we have to check all possible positions of a_j in the plan provided that $a_j < a$.

Step 3.3. The new action a_j is inserted in the position obtained in the previous step. This position may be sequential - between two actions- or parallel to one or more actions. In this latter case, it must be possible to execute all actions in parallel, i.e. none of the actions will require and Add effect of another action or delete any of its preconditions. When this is not possible, actions must be executed sequentially.

The length of the returned plan P' will be the value of the heuristic function $h(x)$. Some of the properties of the heuristic function are:

- if x is an inconsistent or non-reachable state (all literals in x cannot be true at the same time), $h(x)$ returns ∞,
- if x is reachable from $a_0{}'$ so will be the states following x. This helpful information reduces vastly the cost of computing $h(x)$,
- although $h(x)$ is a non-admissible heuristic, it returns the optimal state for most of the test cases in empirical evaluations.

3.2 An Application Example

We will use an example of the *hanoi* domain to illustrate the behaviour of SimPlanner. The problem consists of four disks (huge **H**, big **B**, medium **M** and small **S**) and four pegs **P1**, **P2**, **P3** and **P4**. Initially, the four disks are on **P1** (Figure 4). The goal to be reached is shown in the final state of Figure 4. The unexpected situation occurs after executing the first action in the plan move S M P2 (move disk S from disk M to P2). At this time, a new smaller disk (tiny **T**) appears on **P4**. Figure 5 shows plan Π which is no longer executable because **P4** is not empty any more.

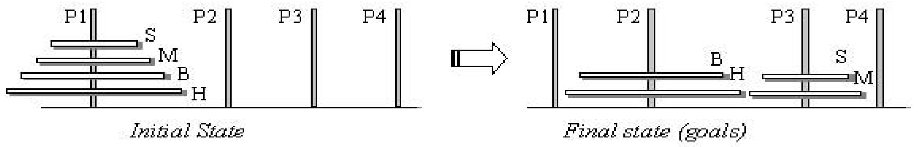


Fig. 4. Initial and final situation in the problem

Literals, denoted by numbers, are represented in Table 1. In Figure 5, literals above each node stand for the action preconditions and literals below each node represent the add and delete effects of the action.

The necessary states for each action in Π are shown in table 2. These are calculated by applying the formula in step 2 of SimPlanner algorithm. Then we compute the set of possible reachable states $RS = \{S(a_1), S(a_2, a_3), S(a_4), S(a_5), S(a_n)\}$.

Table 1. Literals in the *Hanoi* problem

3	on S P2	5	clear P3	6	on H P1	8	on B H	10	on M B	12	clear M
13	on T P4	15	clear S	16	clear T	19	clear P2	20	on M P3	21	clear B
23	clear P4	24	on S M	30	clear H	35	on B P4	39	clear P1	40	on H P2

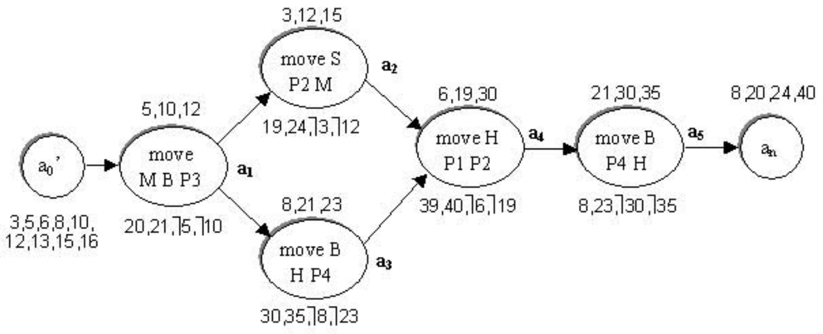


Fig. 5. Remaining plan to be executed (Π)

Notice $S(a_2)$ and $S(a_3)$ form a single state because both actions can be executed at the same time step. The result of $f(x)$ for each possible reachable state is shown in table 3. $h(RS1)$ returns ∞ because it is impossible to satisfy literals 12, 15, 5 and 23 due to the extra disk **T**. The same applies to $RS2$. However, $RS3$ is a reachable state because disk **T** can be located on top of disk **S** and clear S is not a condition required in $RS3$. The selected optimal state is $S(a_4)$ as this is the state that minimizes the value of $f(x)$. In case of equal values for $f(x)$ we will select that state with minimum value of $h(x)$.

Table 2. Necessary states for the hanoi example

Action	Description	Necessary state
a_n	final state	8, 20, 24, 40
a_5	move B P4 H	20, 21, 24, 30, 35, 40
a_4	move H P1 P2	6, 19, 20, 21, 24, 30, 35
a_3	move B H P4	6, 8, 20, 21, 23
a_2	move S P2 M	3, 6, 12, 15, 20, 21
a_1	move M B P3	3, 5, 6, 8, 10, 12, 15, 23

Finally, a plan from $a_0{}'$ to a_4 is built (t0: *current state*, t1: move M B P3, t2: move S P2 M, t3: move T P4 S, t4: move B H P4) and concatenated with the rest of the old plan from action a_4 (t5: move H P1 P2, t6: move B P4 H). For

Table 3. Reachability table for the hanoi problem

x	Pos. reachable state	$h(x)$	$g(x)$	$f(x)$
RS1	$S(a1)$	∞	5	∞
RS2	$S(a2) \cup S(a3)$	∞	3	∞
RS3	$S(a4)$	4	2	6
RS4	$S(a5)$	5	1	6
RS5	$S(a_n)$	6	0	6

our purposes, we have used the planner 4SP [6] as SimPlanner uses most of the data structures which are managed by this planner. The plan obtained when computing $h(x)$ is used as a lower bound in 4SP. For most of the test cases, the plan returned by 4SP was the same as the obtained in the computation of $h(x)$. In the *hanoi* problem, $h(x)$ returns the optimal plan in four execution steps.

4 Experimental Results

SimPlanner has been tested on several domains with different types of input information about external changes. The tested domains are *hanoi*, *monkey*, *blocksworld*, *logistics* and *mobile robots navigation*.

Figure 6 shows the comparative times for several problems between generating a complete plan from scratch or repairing only the affected parts of the plan (replanning process). In all except one of the test cases, the obtained plan was the optimal one. Temporal cost for replanning includes the cost of computing the reachable state plus time for generating the plan. As we can see in figure 6, replanning runtimes are much better when input changes are not very significant. In problems P5 and P7 respectively, the new current state forces to create a complete plan, so the cost of replanning is slightly higher.

We have also made the same tests with planner STAN2000 [7][1]. The time difference between planning and replanning is about the same proportion as the ones shown in Figure 6.

5 Conclusions

SimPlanner is a planning and execution system which allows the user to monitor the execution of a plan, interrupt the monitoring to input new information and repair the plan under execution according to unexpected event. SimPlanner performs an execution monitoring rather than simply testing the next action to execute. In this way, SimPlanner anticipates forthcoming situations and adjusts the plan in accordance.

The key point in SimPlanner is the replanning module. SimPlanner uses a graph-based planning approach supported by heuristic search techniques to efficiently replan in a dynamic world.

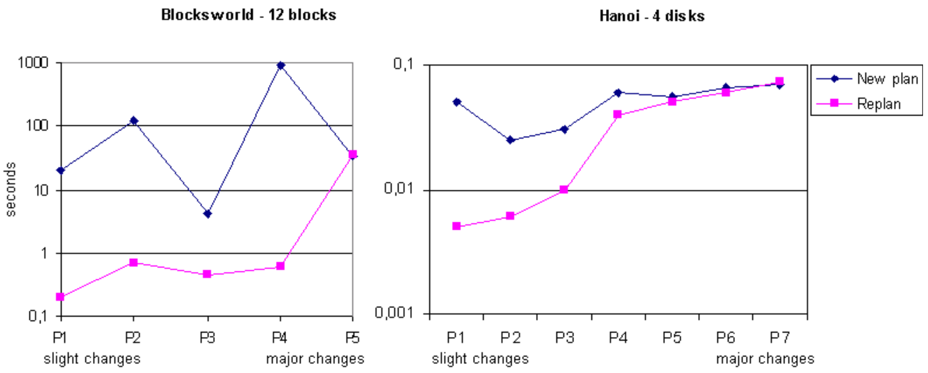


Fig. 6. Comparative results: generating a complete plan versus replanning

Currently, we are working on the integration of the planning algorithm and SimPlanner. The main objective is to be able to guarantee the optimality of the heuristic evaluation and improve the efficiency of the overall process. Additionally, we intend to extend SimPlanner to deal with time and consumable resources (as the battery in robot mobile environments).

References

1. Bacchus F.: AIPS 2000 competition results. Technical Report, University of Toronto, 2000. http://www.cs.toronto.edu/aips2000/.
2. Blum A. L., Furst M.L.: Fast Planning Through Planning Graph Analysis. Artificial Intelligence 90:281–300 (1997).
3. Despouys O., Ingrand F.F.: Propice-Plan: Toward a Unified Framework for Planning and Execution. European Conference on Planning 99, 280–292, 1999.
4. Haslum P., Geffner H.: Admissible heuristics for optimal planning. International Conference on AI Planning and Scheduling, 2000.
5. Hoffmann J., Nebel B.: The FF planning system: fast plan generation through heuristic search. Journal of Artificial Intelligence Research, 14:253–302, 2001.
6. Onaindia E., Sebastia L., Marzal E.: Incremental local search for planning problems. ECAI-2000 workshop on local search for planning & scheduling. Lecture Notes in AI, Springer-Verlag, 2001.
7. Fox M., Long D.: STAN public source code. `http://www.dur.ac.uk/CompSci/research/stanstuff/` (1999)
8. Stone P., Veloso M.: User-guided interleaving of Planning and Execution. Frontiers in Artificial Intelligence and Applications, 103-112, IOS Press, 1996.
9. Wilkins D.E.: Practical Planning: Extending the Classical AI Planning Paradigm. Morgan Kaufmann Publishers, San Mateo, CA, 1988.
10. Wilkins D.E., Myers K.L.: Asynchronous Dynamic Replanning in a Multiagent Planning Architecture. Advanced Planning Technology, 267–274, 1996.

Hybrid Hierarchical Knowledge Organization for Planning

Bhanu Prasad

School of Computer and Information Sciences
Georgia Southwestern State University, Americus, GA 31709, USA
bhanu@canes.gsw.edu

Abstract. This paper presents a hybrid hierarchical knowledge-based system that is currently under development. This system is used for basic planning purposes at present. The important component of the system is a set of plan operator structures. Indexing and packaging hierarchies together play a key role in the construction of a structure. An abstraction hierarchy is used to organize the domain knowledge of the system. The system utilizes some heuristic knowledge of the domain to avoid brittleness. The system has been implemented in the domain of cooking vegetables in Indian style. A restricted natural language interface is incorporated for the convenient specification of user inputs.

1 Introduction

For a given start and goal states, a *plan* is a sequence of operators (or states) that connects the start state to a goal state. The process of finding this sequence is the task of planning. Classical planning approaches are based on search [15], [16], [20], [23]. In contrast, knowledge-based planning approaches depend on past experience for solving new problems. In this paper we call a system as knowledge-based or memory-based if it is based on *Memory Organization Packets* (MOPs) [14]. Either one or more of *abstraction* (i.e., *inheritance*), *indexing* or *packaging* hierarchies [14] are used in implementing a MOP based system. Most of the memory-based planners are based on indexing hierarchies and these systems are referred to as *case-based planning* systems [2], [3], [4], [5], [7], [10], [21]. Some memory-based systems are based on packaging hierarchies [5], [10].

Hierarchical problem solving [4], [5], [10] is an approach using intermediate states, in order to reduce search. Backtracking can be avoided if the intermediate states are generated from memory instead of a trial and error process. In this paper we explore the design of a hierarchical planning system that is based on indexing and packaging hierarchies for organizing planning knowledge. This design is, in some sense, a fruitful combination of the approaches presented in [4], [5], [10]. The plans are tailored to specific requirements, with the aid of an abstraction hierarchy that is used to organize the domain knowledge. The system also utilizes some *heuristic knowledge* of the domain to avoid brittleness. A *restricted natural language interface* (RNLI) is incorporated for the convenient specification of user inputs. Using Common Lisp, the system is implemented in the domain of cooking vegetables in Southern Indian Style. Figure 1 specifies the schematic diagram of the system. The following sections contain some details regarding the system.

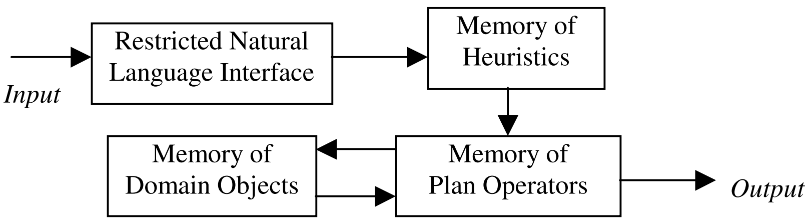


Figure 1. The schematic diagram of the system

2 Restricted Natural Language Interface

A restricted natural language interface is incorporated for the convenient specification of user inputs. The interface is based on a *parser* and a *lexicon* [13], as shown in figure 2. The parser checks each and every word of a specified input and converts it into a set of pre-defined predicates. For example, if the input is: "a fried dish with potato and brinjal" then the output is: "((style fry)(with-vegetable potato)(with-vegetable brinjal))". The output is supplied to the memory of plan operators. Now we discuss the memory organization of the system in detail.

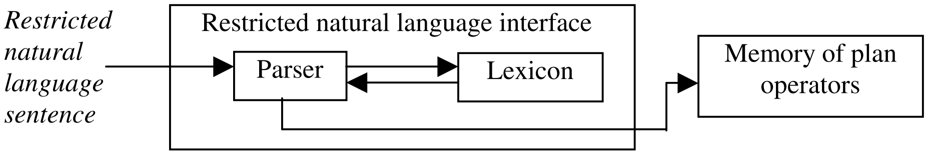

Restricted natural language sentence
Restricted natural language interface
Parser
Lexicon
Memory of plan operators

Figure 2. Restricted natural language interface

3 Organization of the Memory

The domain knowledge of the system is organized using an abstraction hierarchy, as explained in section 3.1. The planning knowledge is organized based on the fact that in structured domains there exists distinct and identifiable classes of plans. The plans belong to the same class are more similar and there are fundamental differences between the plans belong to different categories. In addition to this fact, the system is based on the observation that neither case-based [1], [2], [3], [5], [7], [8], [9], [11], [12], [14], [18], [21], [22] nor script (or, common events)-based [5], [10], [17] approach alone is sufficient in organizing the planning knowledge in some complex structured domains such as ours. In fact, our preliminary results show that a combination of these two approaches is effective in organizing the knowledge. At present, we are evaluating the effectiveness in terms of memory requirements, speed of plan generation, and the flexibility of the system. Planning knowledge organization is explained in section 3.2. To avoid brittleness [19], some heuristic knowledge of the domain is incorporated into the system. The heuristic knowledge organization is explained in section 3.3.

3.1 Memory of Domain Objects

The memory of domain objects organizes the domain knowledge of the system, in our case the cooking domain. It is organized using an abstraction hierarchy. An object, which is either a property or a domain object, is represented as a *frame* [13]. Planning information such as pieces size and cooking time of the ingredients is associated with the objects. This information is retrieved and used by the memory of plan operators while generating a plan. More details of this memory are available in [4], [5], [10].

3.2 Memory of Plan Operators

The system is based on the fact that there are different classes of plans, with the plans belong to the same class are more similar and the plans belong to different classes are basically different.

In structured domains, some times, the experience acquired in the form of cases is get assimilated into more general structures (i.e., generalized cases). This is in accordance with Schank's notion [17] of how people are acquiring scripts. But some times, the knowledge is not get assimilated into general memory structures and remain as independent cases, as we see in case-based reasoning systems. These two different types of knowledge organizations may co-exist within the same system, to handle the planning knowledge. The demonstration of this observation is the primary content of this paper. We also observed that, if there is more similarities between certain episodes (an episode is a case or a sub-case) then the episodes get assimilated into general structures, otherwise they remain as independent cases. In our system, packaging links are used for organizing the general structures and indexing links are used for organizing independent cases.

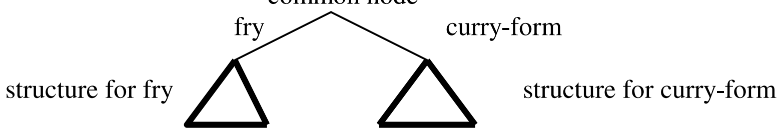


Figure 3. Connection of plan structures to a common root node

In cooking domain, a cooking style represents a class of plans. A style is represented in the form of a tree. The root nodes of different styles are connected to a common node. These links are indexed by the name of the class (i.e., cooking style). Figure 3 contains two structures connected to a common node. In this diagram, the thin lines represent indexing links and the thick triangles represent plan structures (i.e., classes of plans).

A highest-level plan operator identifies a plan structure. The operator in turn has either indexing links or packaging links (but not both) to the *next-level plan*(s) [4], [5], [10]. If the next-level plans are so similar then they are organized using indexing links else they are organized using packaging links. Again, each of the operators in the next-level plan(s) has either indexing or packaging links and so on. Now we discuss the organization of the next-level plan of an operator. Here there are two cases:
First Case: If the operator has indexing links then the next-level plan is in the form of a set of plans that are created in terms of the operators at the next hierarchical level. For example, the operator *fry* has indexing links to the following two plans:
(i). *Basic-preparation*, *prepare-pieces*, *preparation-for-fry*, *perform-fry* (corresponds to the ingredient *lady's finger*) (ii). *Basic-preparation*, *boiling*, *preparation-for-fry*, *perform-fry* (corresponds to the ingredient *peas*).
These plans are the next-level plans for the operator *fry*.

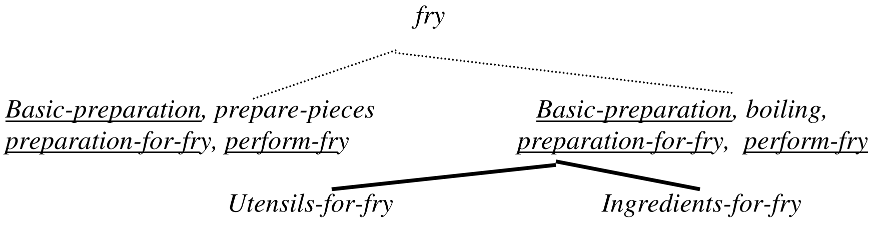

fry
Basic-preparation, prepare-pieces
preparation-for-fry, perform-fry
Basic-preparation, boiling,
preparation-for-fry, perform-fry
Utensils-for-fry
Ingredients-for-fry

Figure 4. Internal organization of a structure

Second Case: If the operator has packaging links then the next-level plan is in the form of a sequence of operators that will be present in any of the expansions of that operator. For example, the operator *preparation-for-fry*, presented in the previous example, has packaging links to the following operators: *utensils-for-fry*, *ingredients-for-fry*. These two operators (in the specified order) together constitute the next-level plan of the operator *preparation-for-fry*.

Common operators are shared between different plans. Figure 4 depicts a portion of the plan structure corresponding to the cooking style fry. In this figure, dotted lines represent indexing links and solid arcs represent packaging links. In addition, common operators are underlined and the indexes are not shown in the figure.

If an operator has indexing links then the operator may have more than one next-level plans. One of the next-level plans will be adopted while generating a plan. The modified plan is stored back into the memory. In the case of packaging links, the operator has exactly one next-level plan and this plan is always being adopted while generating a plan. The modified plan won't be stored back in this case.

3.3 Memory of Heuristics

Heuristic knowledge is used to avoid the *brittleness* [19] of the system. To translate user requirements into plan specifications, some common heuristics that are valid across the spectrum of plans are incorporated into the system. For example, in cooking domain, a fried dish with little amount of *oil* needs to have its ingredients chopped in smaller pieces. This knowledge is organized in the form of if-then rules. The memory of heuristics is a *forward chaining rule-based system* [6]. The *fact part* contains the properties of the domain objects and the input supplied by RNLI. The *rule part* contains plan (i.e., recipe) independent if-then rules. The rule-based system generates inferences based on the input and supplies it to the memory of plan operators for plan generation. Now we see the planning algorithm.

4 Planner

The RNLI accepts user input and converts it into a set of pre-defined predicates. The output is supplied to the memory of heuristics, where more predicates may be added to the set by the rule-based system. The resultant set is supplied to the memory of plan operators. Based on the class specified in the input, a suitable plan structure is selected for expansion. A plan is generated by expanding the structure in a hierarchical fashion, with the aid of the memory of domain objects. The actual algorithm is presented below.

1. /* RECEIVE INPUT */ Accept user specifications, supplied using the RNLI. Converts it into a set of predicates.

2. /* REFINE INPUT */ Refine the input using the rule-based system in the memory of heuristics and supply the resultant set to the memory of plan operators.

3. /* SELECT STRUCTURE */ Select the operator that represents the root node of the required structure.

4. /* INITIALIZE */ For each level *i* in the hierarchy of the structure, create a list *plan-list-i* to store the operator identifiers at that level. Initialize each list to NULL. Add root node to *plan-list-0*. Set $i \leftarrow 0$. Retrieve planning information from the memory of domain objects

5. /* EXPAND, MODIFY AND POSSIBLE UPDATE */ For each of the operators (from left to right) in *plan-list-i*, do the following in the specified order.

(a). If it is a high-level operator then
(i). Generate the next-level plan using the indexing or packaging links. (ii). Modify the plan using the appropriate modification rules associated with the

operator. (iii). If the indexing links are used for plan generation then store the modified plan.

(b). If the operator is a ground-level one then

(i). If the operator has a variable and is not instantiated then compute the value of the variable. Instantiate the variable with the value. (ii). If either the template doesn't have a variable or all the variables in the template are already instantiated then leave the operator unchanged.

(c). Append the modified next-level plan or the ground-level operator to *plan-list-(i+1)*.

6. /* DESCEND */ If any of the elements in *plan-list-(i+1)* is not an instantiated ground-level one then set $i \leftarrow (i+1)$ and go to step 5. Else output the ground-level plan.

5 Comparison and Concluding Remarks

In this paper we have presented an approach for organizing knowledge using hybrid hierarchical structures. We also demonstrated how this knowledge organization is useful for basic planning purposes. This system has used MOP-based hierarchies for knowledge organization. In addition, we have incorporated a rule-based expert system for avoiding brittleness. The system is currently under development. As a result, the performance results such as memory requirements, plan generation speed, and flexibility are not presented in this paper.

There are some similarities between this system and ABSTRIPS [15]: both are hierarchical. Unlike ABSTRIPS, in our system, there are no hierarchies among the preconditions of the operators. Instead, the hierarchies are among the operators itself. The system is fundamentally different from other classical systems [16], [20], [23], which generate plans from scratch. In our system there is already a partial plan, in the form of packaging and indexing links. As a result, generating a plan in our system is much more easier when compared to classical systems.

There is some similarity between our system and MEDIATOR [18] that uses pieces of cases in creating a case. MEDIATOR has to select a case in order to access the relevant part. Where as in our system, a complete case is never accessed in one shot.

Even though both are hierarchical, the difference between our system and CELIA [12] is that in CELIA, a link represents a relationship between goals in a case. In our system, a link connects an operator and its next-level plan(s).

There is a similarity between our system and the case-based planning system CHEF [7]: both operate in cooking domain. But there are some fundamental differences between CHEF and our system. CHEF is organized completely based on indexing hierarchies, where as in our system, a combination of indexing and packaging hierarchies are used for memory organization. CHEF can repair its failed plans and there is no such mechanism yet in our system. Finally, CHEF does not have the memory of domain objects to guide its planning process.

PRIAR [9] uses the validation structure of a plan, which represents the dependencies among the plan steps, for retrieval and reuse purposes. Our system uses a structure for the purpose of plan generation. In addition, unlike PRIAR, our system does not have backtracking facility for undoing a wrong step. In stead, our system utilizes the memory of domain objects to guide the expansion of a structure.

The system is different from PRODIGY/ANALOGY [21], which is based on *generative planning* and case-based reasoning. In this system, the retrieved solution is not used for adaptation. Instead it is used to guide a classical nonlinear planner, in generating the solution. In our system, the retrieved case is used for adaptation. The modification knowledge and the memory of domain objects are used for this purpose.

There are some similarities between the current system and the systems presented in [4], [5], [10]. All these systems are implemented in the cooking domain. They are all based on the concept that, in structured domains, there are different classes of plans. But this system is different because the system presented in [5], [10] is based on packaging links to organize plans. Similarly, the system presented in [4] is based on indexing links to organize the planning knowledge. But the present system is based on both indexing and packaging links in organizing a class of plans. As a result, this system is (hoped to be) more flexible in handling different types of planning knowledge. The previous systems force us to use a particular structure, either indexing or packaging. In addition, the systems presented in [4], [10] do not have any mechanism to avoid brittleness.

There is a similarity between PARIS [2] and our system: both are case-based and both organize cases at different levels of abstraction. But there are differences too: in order to solve a new problem, PARIS retrieves an abstract case, whose abstract problem description matches with the current problem description. A generative planner that performs a forward directed state space search modifies the abstract case. In our system, a suitable structure is adopted and hierarchically modified, with the aid of the modification rules that are associated with the operators.

Now we are in the process of incorporating a learning component into the system, so that the system can learn the modification rules, from a sufficient set of cases. We are making our system more efficient by making it learn from previous failures. In addition, we are in the process of developing a mechanism for repairing failed plans.

References

1. Bergmann, R., Breen, S., Göker, M., Manago, M., Wess, S.: Developing Industrial Case-Based Reasoning Applications: The INRECA Methodology. Springer-Verlag, NY (1999).
2. Bergmann, R., Munoz, H., Veloso, M., Melis, E.: Case-Based Reasoning Applied to Planning Tasks. Lenz, M., Bartsch-Spörl, B., Burkhard, H.D., Wess, S. (eds.). Case-Based Reasoning Technology from Foundations to Applications, Springer-Verlag, Berlin (1998).
3. Bergmann, R., Wilke, W.: Building and Refining Abstract Planning Cases by Change of Representation Language. Journal of AI Research (1995).
4. Bhanu Prasad: Planning with Case-Based Structures. AAAI Fall Symposium on Adaptation of Knowledge for Reuse Aha, D., Ram, A. (eds). AAAI Press (1995).
5. Bhanu Prasad: Planning With Cooperative Memory Structures. Ph.D. Thesis, Department of Computer Science and Engineering, Indian Institute of Technology, Madras (1998).
6. Golshani, F.: Rule-Based Expert Systems. Knowledge Engineering: Fundamentals. Adeli, H. (editor). McGraw-Hill, NY (1990).
7. Hammond, K.J.: Case-Based Planning, Viewing Planning as a Memory Task. Academic Press, NY (1989).
8. Leake, D.B.: Case-Based Reasoning: Experiences, Lessons, and Future Directions. AAAI Press, CA (1996).
9. Kambhampati, S.: A Theory of Plan Modification. Proceedings of AAAI (1990).
10. Khemani, D., Bhanu Prasad: A Memory-Based Hierarchical Planner. Veloso, M., Aamodt, A. (eds). Case-Based Reasoning Research and Development, Springer-Verlag, NY (1995).
11. Kolodner, J.: Case-Based Reasoning. Morgan Kaufmann, CA (1993).
12. Redmond, M.: Learning by Observing and Understanding Expert Problem Solving. Ph.D. Thesis, Georgia Institute of Technology (1992).
13. Rich, E., Knight, K.: Artificial Intelligence, Second Edition, McGraw-Hill, NY (1991).
14. Riesbeck, C.K., Schank, R.C.: Inside Case-Based Reasoning, Lawrence Erlbaum Associates, NJ (1989).
15. Sacerdoti, E.D.: Planning in a Hierarchy of Abstraction Spaces. Journal of Artificial Intelligence, 5(2) (1974).
16. Sacerdoti, E.D.: A Structure for Plans and Behavior. Elsevier, NorthHolland (1977).
17. Schank, R.C., Abelson, R.P.: Scripts, Plans, Goals, and Understanding. Lawrence-Erlbaum, NJ (1977).
18. Simpson, R.L.Jr.: A Computer Model of Case-Based Reasoning in Problem Solving. Ph.D. Thesis, Georgia Institute of Technology (1985).
19. Sun, R.: Robust Reasoning: Integrating Rule-Based and Similarity-Based Reasoning, Journal of Artificial Intelligence, 75 (1995).
20. Tate, A.: Project Planning Using a Hierarchic Non-Linear Planner, Research Report No. 25, Department of Artificial Intelligence, University of Edinburgh (1975).
21. Veloso, M.M.: Planning and Learning by Analogical Reasoning. Springer-Verlag, NY (1994).
22. Watson, I.: Applying Case-Based Reasoning: Techniques for Enterprise Systems. Morgan Kaufmann, CA (1997).
23. Wilkins, D.E.: Practical Planning-Extending the Classical AI Planning Paradigm. Morgan Kaufmann, CA (1988).

STeLLa: An Optimal Sequential and Parallel Planner*

L. Sebastia, E. Onaindia, E. Marzal

Dpto. Sistemas Informaticos y Computacion
Universidad Politecnica de Valencia
e-mail: {lstarin, onaindia, emarzal}@dsic.upv.es

Abstract. In the last few years, the field of planning in AI has experimented a great advance. Nowadays, one can use planners that solve complex problems in a few seconds. However, building good quality plans has not been a main issue. In this paper, we introduce a planning system whose aim is obtaining the optimal solution w.r.t. the number of actions and maintaining as maximum number of parallel actions as possible.

1 Introduction

The field of planning in AI has experimented great advances over the last few years. However, most efforts have been devoted to the development of fast planners [8] [1], capable of solving a wide range of problems. This fact has focused researchers' attention on the development of heuristics to accelerate the planning process, considering less important the optimization of the obtained plans. However, in real application environments, apart from finding efficient planning processes, it is important to obtain (close to) optimal plans in order to reduce their execution cost.

In general terms, current planning systems can be classified into:

- **Sequential planners**, like FF [5], HSP [3] and Seristar [4]. The plan is a totally ordered sequence of actions. The optimal plan is the shortest one. Seristar guarantees this optimal solution, unlike FF, which is much faster.
- **Parallel planners**, like STAN [7] and Parastar [4]. The plan is a totally ordered sequence of time steps in which a set of actions can be executed. The optimal solution consists of the plan with the minimum number of time steps. Both STAN and Parastar are able to find this solution.

In this paper, we present STeLLa[1], a planner that, assuming a constant cost for each action, builds the minimal cost plan and then, reduces the number of time steps by executing some actions in parallel. The main novel issue in STeLLa algorithm, which implements a forward chaining search, is the use of *landmarks* graphs.

* This work has been partially supported by the project n. 20010017 - *Navigation for Autonomous Mobile Robots* of the Universidad Politécnica de Valencia

[1] STeLLa is an acronym of the words Sequential, TimE Layer and LAndmarks.

The concept of landmarks graph (LG) is introduced in [9], where a landmark is defined as a literal that has to be achieved in every solution plan. The process extracts a set of landmarks that are ordered under the concept of "reasonable order", obtaining a LG. In [10] the idea of "reasonable order" is extended. Unlike these previous approaches, STeLLa uses the concept of LG to determine at each point the next set of literals to satisfy and the best actions to execute in order to achieve this set of literals.

This paper is organized as follows. In section 2 some basic concepts about LGs are given. In section 3 the STeLLa algorithm is described and an example is shown in section 4. Section 5 gives the results obtained so far and section 6 concludes by summarizing the strong and weak points of this algorithm.

2 Foundations

This section gives some basic concepts to understand the structure of a LG.

The process to build a LG consists of two steps [10]: (1) extracting the set of landmarks and (2) ordering the obtained set of landmarks.

Definition 1. *Given a planning task* $\mathcal{P} = (\mathcal{O}, \mathcal{I}, \mathcal{G})$*, a fact* l_i *is a* **landmark** *in* $\mathcal{P}$ *iff* l_i *is true at some point in all solution plans, i.e., iff for all* $P = \langle o_1, \ldots, o_n \rangle, \mathcal{G} \subseteq Result(\mathcal{I}, P) : l_i \in Result(\mathcal{I}, \langle o_1, \ldots, o_i \rangle)$ *for some* $0 \leq i \leq n$*. The* **side-effects** *of a landmark* l_i *are defined as: side_effects*$(l_i) := \{\mathsf{Add}(o) - \{l_i\} \mid o \in \mathcal{O}, l_i \in \mathsf{Add}(o)\}$

The extraction process is straightforward. First, a relaxed planning graph (RPG) is built. Then, a backward search is performed over this RPG, selecting as landmarks all the literals that belong to the intersection of the preconditions of the actions achieving a top level goal or a landmark. The remaining preconditions are grouped into a disjunctive set.

Definition 2. *Let* l_i, l_j *be two landmarks.* l_i *and* l_j *are* **consistent** *if* $\exists$ *a possible state* $S / l_i \in S \wedge l_j \in S$*.*

We use the TIMinconsistent function [2], which returns whether two literals are consistent or not. Once the set of landmarks has been extracted, they are ordered according to the following orders, which in turn define an LG:

Definition 3. *A* **natural order** *is established between two landmarks* l_i, l_j $(l_i <_n l_j)$ *when in every solution plan is necessary to solve* l_i *to achieve* l_j*, that is,* l_i *is a precondition of all the actions that satisfy* l_j*.*

Definition 4. *We establish a* **weakly reasonable order** *in the following cases:*

- *landmarks* l_i *and* l_j *that are naturally ordered before the same node* l_k *can be ordered* $l_i <_{wr} l_j$*, if* $\exists$ *landmark* $x : x <_n l_i \wedge \mathsf{TIMinconsistent}(x, l_j) = \mathsf{TRUE}$
- *two landmarks* l_i *and* l_j *can be ordered* $l_i <_{wr} l_j$ *if there exists some other landmark* x*, and* x *and* l_j *are ordered before the same node; and there is an ordered sequence of* $<_n$ *orders that post* l_i *before* x*. In this situation,* l_i *and* l_j *are weakly ordered if* $\exists$ *landmark* $y : y <_n l_i \wedge \mathsf{TIMinconsistent}(y, l_j) = \mathsf{TRUE}$

```
Algorithm STeLLa ($\mathcal{I},\mathcal{G}$) $\rightarrow$ plan $\mathcal{P}$ organized in time steps
Build $LG(\mathcal{I},\mathcal{G})$
$\mathcal{C}=\mathcal{I}$; time = 0
While $\mathcal{G} \not\subset \mathcal{C}$
    1. Compute fringe $\mathcal{F}$
    2. Select a consistent subset $\mathcal{CS}$ of landmarks from $\mathcal{F}$
    3. Compute the set of actions $\mathcal{A}$ that solve $\mathcal{CS}$
    4. Select the actions from $\mathcal{A}$ that have to be executed
    5. If $\mathcal{A}=\emptyset$, build $LG(\mathcal{C},\mathcal{G})$
         Else Execute $\mathcal{A}$, obtaining the new $\mathcal{C}$
              $\mathcal{P}_{time}=\mathcal{A}$; time = time + 1
```

Fig. 1. STeLLa algorithm

– *a pair of landmarks l_i and l_j is ordered $l_i <_{wr} l_j$ if $\exists$ landmark x : $l_i <_n x \wedge l_j <_{wr} x \wedge \mathsf{TIMinconsistent}(side\_effects(l_j), l_i) = \mathsf{TRUE}$*

Definition 5. *A* **LG** *is a graph (N, E) where:*

– *$l_i \in N$ if l_i is a landmark or a disjunctive set*[2].
– *$l_i, l_j \in N, (l_i, l_j) \in E$ if $l_i <_n l_j \vee l_i <_{wr} l_j$.*
– *Moreover, the following information extracted from side-effects is added: if l_i is a side-effect of l_j and viceversa, a conjunctive set $[l_i, l_j]$ is built, so that $[l_i, l_j]$ is added to N and $\forall l_k / (l_k, l_i) \in E \vee (l_k, l_j) \in E \rightarrow E = E \cup (l_k, [l_i, l_j])$*

Because the process for extracting landmarks and the computation of orders are approximated computations, the information in the LG can be incomplete.

Definition 6. *Let a_i, a_j be two actions:*

– *a_i causes a* **NF1** *conflict over a_j if $\exists d \in \mathsf{Del}(a_i) /\ d \in \mathsf{Pre}(a_j) \wedge a_i < a_j$*
– *a_i and a_j are in* **NF2** *conflict if $\exists d_i \in \mathsf{Del}(a_i) /\ d_i \in \mathsf{Pre}(a_j) \wedge \exists d_j \in \mathsf{Del}(a_j) /\ d_j \in \mathsf{Pre}(a_i)$*
– *a_i causes a* **F** *conflict over a_j if $\exists d \in \mathsf{Del}(a_i) /\ d \in \mathsf{Pre}(a_j) \vee d \in \mathsf{Add}(a_j)$*

3 Algorithm

The algorithm implemented in STeLLa is shown in Figure 1. Each point in this algorithm is explained in the following sections.

3.1 Computation of $\mathcal{F}$

In each iteration, STeLLa computes the set of landmarks to be solved at the current time step, called **fringe** and denoted by $\mathcal{F}_{time}$. Let $LG(N, E)$ be the

[2] For the sake of simplicity, we call *landmarks* all the elements in N throughout the rest of the paper.

current LG and $l_i \in N$, $l_i \in \mathcal{F}_{time}$ if $\forall l_j \in N/(l_j, l_i) \in E, l_j \in \mathcal{F}_{time'}, time' < time$. From now on, we will identify $\mathcal{F}_{time}$ as $\mathcal{F}$.

However, due to the incompleteness in the LG, it might be the case that not all the landmarks in $\mathcal{F}$ can be achieved right now. Therefore, it is necessary to check whether a literal has to be postponed. In first place, for disjunctive sets, only natural orders have been computed during the process for building a LG. Therefore, a disjunctive set l_i belonging to a $LG(N, E)$ is added to the fringe $\mathcal{F}$ if $\forall l_j \in N/(l_i, l_j) \in E, \forall l_k \in N/(l_k, l_j) \in E \rightarrow l_k \in \mathcal{F}_{time'}, time' <= time$. On the other hand, a landmark might be delayed because its producer actions remove a literal which is required later in the planning process. Let $LG(N, E)$ be a LG and f be a landmark in its corresponding $\mathcal{F}$:

1. $P = \{l_i/(l_i, f) \in E \ \wedge \ \mathsf{TIMinconsistent}(l_i, f) = \mathsf{TRUE}\}$
2. if $\exists p \in P/out-degree(p) > 1 \wedge \exists l \in N/l \notin \mathcal{F} \wedge (p, l) \in E \ \wedge$ $\mathsf{TIMinconsistent}(l, p) = \mathsf{FALSE} \rightarrow \mathcal{F} = \mathcal{F} - f$

3.2 Computation and selection of consistent subsets

Due to incompleteness in the LG, it may be the case that not all the landmarks in $\mathcal{F}$ can be in the same state, that is, they are inconsistent. Therefore, the first step in order to ensure that the literals in $\mathcal{F}$ can be achieved *at the same time* from the current state is building consistent subsets from $\mathcal{F}$.

Definition 7. *Let $LG(N, E)$ and $\mathcal{F}$ be the current LG and fringe. A $\mathcal{CS}_i$ is a set of literals where* $\forall l_j \in \mathcal{CS}_i, l_j \in \mathcal{F} \wedge \nexists l_k \in \mathcal{CS}_i/\mathsf{TIMinconsistent}(l_j, l_k) = \mathsf{TRUE}$

Once $\mathcal{F}$ has been divided into consistent sets $\mathcal{CS}_i$, we select one of them, called $\mathcal{CS}$, which will determine the subset of landmarks to be satisfied at this iteration. Therefore, we evaluate each $\mathcal{CS}_i$ under the following criteria and select the one with the best value:

1. We compute an approximation to the new state $\mathcal{C}'$ in the following way: $\mathcal{C}' = \mathcal{CS}_i \cup \{f_j \in \mathcal{C}/\forall f_i \in \mathcal{CS}_i, \mathsf{TIMinconsistent}(f_i, f_j) = \mathsf{FALSE}\}$
2. Next step is to build the LG for $\mathcal{C}'$ and evaluate it by using the same criteria that will be applied in the following section for the action selection.

3.3 Computation and selection of actions

For each literal l_i in $\mathcal{CS}$, the set of applicable actions $\mathcal{A}_i$ is computed according to the type of l_i:

1. Literal: $\mathcal{A}_i = \{o \in \mathcal{O}/l_i \in \mathsf{Add}(o)\}$
2. Conjunctive set: $\mathcal{A}_i = \{o \in \mathcal{O}/l_i \subseteq \mathsf{Add}(o)\}$
3. Disjunctive set: $\mathcal{A}_i = \bigcup_{f_j \in l_i} \{o \in \mathcal{O}/f_j \in \mathsf{Add}(o)\}$

We define $\mathcal{A}$ as a set containing the actions to be executed in the current iteration. In order to build this set, only one action from each $\mathcal{A}_i$ is selected by using this heuristic function $h(a_{ij}) = f(\mathsf{applicable}, \mathsf{NF1}, \mathsf{NF2}, \mathsf{LG\text{-}correct}, \mathsf{LG\text{-}delete})$, where:

1. applicable$(a_{ij}) = 1$ if a_{ij} can be executed in the current state and its execution does not generate a previously reached state; 0, otherwise.
2. NF1$(a_{ij}) = 1$ if a_{ij} produces a NF1 conflict with all the landmarks immediately following $\mathcal{F}$; 0, otherwise.
3. NF2$(a_{ij}) = 1$ if a_{ij} generates a NF2 conflict with every action in another set $\mathcal{A}_k$; 0, otherwise.
4. Build the LG corresponding to the state resulting from the application of a_{ij} in the current state. LG-correct$(a_{ij}) = 1$ if none of the actions that would solve the set $\mathcal{F}$ of this LG would generate a NF1 or NF2 conflict; 0, otherwise.
5. LG-delete(a_{ij}) is the number of landmarks that would be necessary to reachieve after applying a_{ij}, i.e., the number of delete effects of a_{ij} which are required again as preconditions in the LG.

However, some conflicts may arise among the actions in $\mathcal{A}$ and then some actions must be postponed. For each pair of actions $a_i, a_j \in \mathcal{A}$, if a_i and a_j produce a NF2 conflict, two new sets are built by removing the landmarks that a_i and a_j achieve respectively, and one of them is selected according to the selection criteria for consistent sets; otherwise, if a_i produces a F conflict over a_j, a_i is ruled out from $\mathcal{A}$.

4 Application example

In this section the behaviour of the STeLLa algorithm is shown through an example whose goal is moving two objects: object O1 is in a secure box and object O2 is grabbed by one robot at $l5$ and there is another robot at $l1$. The plan to solve this problem entails to take a key at $l4$ and push an elevator from $l2$ to $l3$ where the secure box is, then go up with the elevator to reach the secure box and open it with the key for grabbing O1; carry O2 to $l3$ and drop it there. Not all locations are connected among themselves: it is not possible to move directly from $l3$ to $l4$ or from $l1$ to $l3$, and from $l5$ it is only possible to reach $l3$.

In the LG for the initial state (Fig. 2), $\mathcal{F}$ = {*at-robot R1 L4, at-robot R1 L2, at-robot R2 L3*}. Not all the literals in $\mathcal{F}$ are consistent, so two subsets are built: $\mathcal{CS}_1$ = {*at-robot R1 L2, at-robot R2 L3*}, $\mathcal{CS}_2$ = {*at-robot R1 L4, at-robot R2 L3*}. Then we evaluate these sets by building the corresponding LGs (Figs. 3 and 4). We choose $\mathcal{CS}_2$ because no conflicts are found whereas in the LG corresponding to $\mathcal{CS}_1$ a NF2 conflict is generated between the actions that would solve the landmarks *(at-robot R1 L4)* and *(at E1 L3)*. The executable actions for solving each of the landmarks in this set are Go-to R1 L1 L4 y Go-to R2 L5 L3. Now, the new $\mathcal{F}$ (Fig. 4) is $\{(at{-}robot\ R2\ L2{+}at{-}robot\ R1\ L2), at\ O2\ L3, has{-}key\ K1\ R1, at-robot\ R1\ L2\}$. Since no more conflicts arise, the selected actions are: Get-key R1 L4 K1, Drop R2 L3 O2, Go-to R1 L4 L2. However, actions Get-key R1 L4 K1 and Go-to R1 L4 L2 produce a F conflict, and then the latter is postponed to the next time step. From this moment on, the process is similar, obtaining the following plan: Lv_1= Go-to R1 L1 L4, Go-to R2 L5 L3, Lv_2=Get-key R1 L4 K1, Drop R2 L3 O2, Lv_3=Go-to R1 L4 L2, Lv_4=Push-elevator R1 L2 L3 E1, Lv_5=Climb R1 L3 E1, Lv_6=Get-from-secure-box R1 L3 O1 E1 K1.

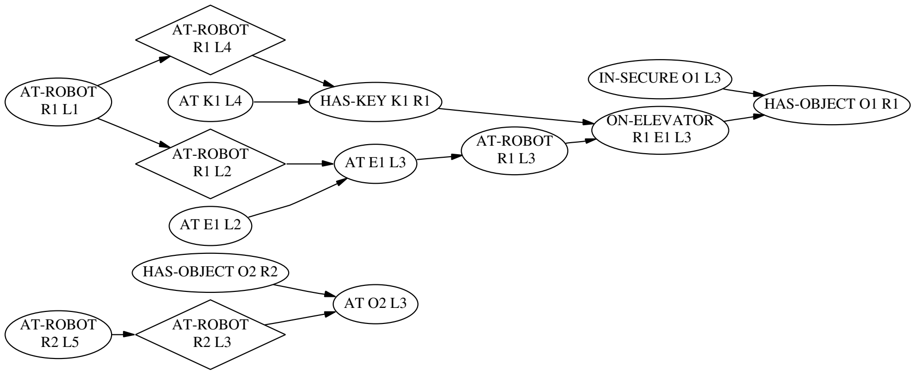


Fig. 2. Initial situation for the `robot` problem

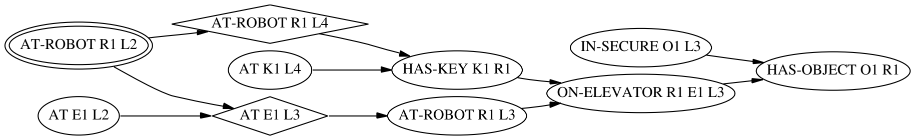


Fig. 3. A part of the LG corresponding to $\mathcal{CS}_1$ in the `robot` problem

5 Experiments

Tables 1 and 2[3] show a comparison between the results obtained with STeLLa and other planners. Table 1 shows that, for sequential and `logistics` domain problems, there are only few differences among all the planners. For example, in *huge* and *robot_connected_t1* problems, FF returns plans longer than necessary; in `logistics`, Parastar obtains the optimal solution in time steps, but the number of actions is clearly larger than necessary. In *prob05* and *ptia1*, STeLLa optimizes the number of total actions against STAN which optimizes the number of time steps. In *loga* problem, both STeLLa and STAN do not obtain the optimal solution. As for problems in the `freecell` domain, it is remarkable the fact that, although STeLLa cannot obtain the optimal solution in *3-1*, it clearly outperforms FF and STAN in problems from *4-1* on. In `miconic`, STAN obtains slightly longer plans than FF and STeLLa.

Execution times are not shown because, as we explained in the introduction, the main objective of STeLLa is improving plan quality, and therefore, performance is not a main issue. However, it must be said that STeLLa's performance is comparable with STAN's in most of the problems. On the other hand, the performance of both Seristar and Parastar are worse than STeLLa .

[3] It has been impossible to execute Seristar and Parastar over `Freecell` and `Miconic` domains

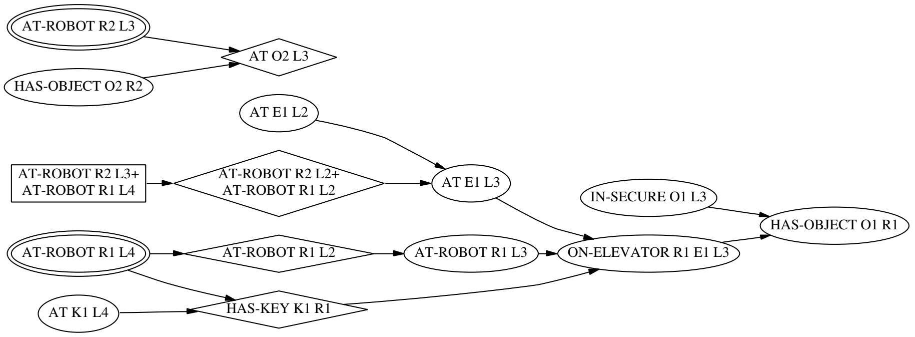


Fig. 4. LG corresponding to $\mathcal{CS}_2$ in the `robot` problem

Table 1. Comparison in sequential and `logistics` domain problems. Results given in `number of actions` or `time steps/actions`.

Problem	**Seristar**	**FFv2.2**	**Parastar**	**STANv3 (2000)**	**STeLLa**
Blocks_Tw_rever10	20	20	20/20	20/20	20/20
Blocks_Tw_diff_10	36	36	36/36	36/36	36/36
Blocks_Huge	18	24	18/18	18/18	18/18
Blocks_Huge2	10	12	10/10	10/10	10/10
Robot_connected_t1	6	7	6/6	(6/6)	6/6
Robot_connected_t2	6	6	6/6	(6/6)	6/6
log_prob02	15	15	9/22	9/15	9/15
log_prob04	15	15	9/20	9/15	9/15
log_prob05	9	9	6/10	6/10	8/9
log_prob06	-	25	11/41	11/25	11/25
log_ptia1	-	24	10/44	(10/25)	11/24
log_loga	-	52	-	(11/54)	14/53

6 Conclusions and further work

We have presented STeLLa, a planner that obtains the optimal solution w.r.t. the number of actions allowing parallelism for most of the solved problems.

The weak point of the algorithm is its high dependency on the LG. Due to the information in the LG can be quite incomplete, some difficulties may arise during the plan construction. This is the case for those problems for which STeLLa has been unable to achieve the optimal solution. In these situations, the LG has to be deeply analyzed in order to find further constraints between landmarks. Also, other techniques could be used to extract the necessary information. In this sense, it would be very helpful to find new criteria to be used for the consistent subsets and action selection. Criteria, such as distance to the goal state, might reduce the number of action combinations in each $\mathcal{A}_i$, thus improving STeLLa performance.

Table 2. Comparison in `freecell` and `miconic` domains. Results for STAN in brackets indicates that the version used is 2000.

Problem	FF	STAN	STeLLa	Problem	FF	STAN	STeLLa
freecell-2-1	9	6/9 (9)	6/9	mic-s9-0	31	(31)	26/31
freecell-2-2	8	6/8 (8)	6/8	mic-s11-0	37	(37)	31/37
freecell-2-3	9	5/9 (8)	5/9	mic-s13-0	44	(44)	36/44
freecell-2-4	9	6/10 (8)	6/9	mic-s15-0	46	(47)	33/46
freecell-2-5	9	6/9 (10)	6/9	mic-s16-0	53	(54)	42/53
freecell-3-1	21	7/15 (16)	9/16	mic-s17-0	56	(59)	44/56
freecell-3-2	20	7/14 (16)	7/14	mic-s18-0	59	(59)	46/59
freecell-4-1	26	(24)	12/21	mic-s19-0	62	(63)	48/62
freecell-4-2	26	(22)	9/19	mic-s20-0	64	(65)	48/64
freecell-5-1	37	(34)	12/27	mic-s21-0	70	(70)	56/70
freecell-5-2	33	(34)	14/27	mic-s22-0	73	(74)	58/73
freecell-6-1	43	(42)	20/34	mic-s23-0	76	(79)	60/76

As a conclusion, and taking into account this is the first version of STeLLa, we consider the results are very promising.

References

1. F. Bacchus. AIPS-2000 competition results. Technical report, University of Toronto, 2000. http://www.cs.toronto.edu/aips2000/.
2. M. Fox and D. Long. The automatic inference of state invariants in TIM. *Journal of Artificial Intelligence Research*, 9:367–421, 1998.
3. B. Bonet and H. Geffner. Planning as Heuristic Search. *Artificial Intelligence*, 2001, forthcoming.
4. P. Haslum and H. Geffner. Admissible heuristics for optimal planning. In *International Conference on AI Planning and Scheduling*, 2000.
5. J. Hoffmann and B. Nebel. The FF planning system: Fast plan generation through heuristic search. *Journal of Artificial Intelligence Research*, 14:253–302, 2001.
6. J. Koehler, B. Nebel, J. Hoffmann, and Y. Dimopoulos. Extending planning graphs to an ADL subset. In *Fourth European Conference in Planning*, pages 273–285. Springer LNAI 1348, 1997.
7. D. Long and M. Fox. Efficient implementation of the plan graph in stan. *Journal of Artificial Intelligence Research*, 10:87–115, 1999.
8. D. McDermott. AIPS-98 competition results, 1998. ftp://ftp.cs.yale.edu/pub/mcdermott/aipscomp-results.html.
9. J. Porteous and L. Sebastia. Extracting and ordering landmarks for planning. In *19th Workshop of the UK Planning and Scheduling Special Interest Group*, 2000.
10. J. Porteous, L. Sebastia, and J. Hoffmann. On the extraction, ordering, and usage of landmarks in planning. In *Recent Advances in AI Planning. 6th European Conference on Planning (ECP'01)*. Springer Verlag, 2001.

Author Index